AML Summary

Lukas Diebold

January 5, 2026

Contents

1 Math Preliminaries

Gibbs Distribution The Gibbs distribution is a probability distribution that assigns likelihoods to states based on a cost function, with lower-cost states being more probable. Given a set of states $x \in \mathcal{X}$, a cost function $E(x)$, and an inverse temperature parameter $\beta > 0$, the Gibbs distribution is

$$p(x) = \frac{1}{Z} e^{-\beta E(x)}$$

where the partition function Z ensures normalization

$$Z = \sum_{x \in X} e^{-\beta E(x)}$$

1.1 Conditional distribution of a multivariate Gaussian

Let

$$\begin{bmatrix} a \\ b \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_a \\ \mu_b \end{bmatrix}, \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix} \right),$$

where Σ_{aa} and Σ_{bb} are covariance blocks and $\Sigma_{ab} = \Sigma_{ba}^\top$. Then the conditional distribution of a given b is again Gaussian with

$$\mathbb{E}[a \mid b] = \mu_a + \Sigma_{ab} \Sigma_{bb}^{-1}(b - \mu_b), \quad \mathrm{Cov}(a \mid b) = \Sigma_{aa} - \Sigma_{ab} \Sigma_{bb}^{-1} \Sigma_{ba}.$$

This identity underlies Gaussian process prediction, Bayesian linear regression posteriors, and Kalman filtering.

Derivation. Write the joint as a block Gaussian and use the Schur complement. The joint log-density (up to constants) is

$$\ell(a,b) = \tfrac{1}{2} \begin{bmatrix} a - \mu_a \\ b - \mu_b \end{bmatrix}^\top \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix}^{-1} \begin{bmatrix} a - \mu_a \\ b - \mu_b \end{bmatrix}.$$

Completing the square in a (or applying the standard conditional Gaussian formula) yields

$$\mathbb{E}[a \mid b] = \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(b - \mu_b), \quad \mathrm{Cov}(a \mid b) = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}.$$

Equivalently, these follow from the block inversion identity and the Schur complement of Σ_{bb}.

1.2 Schur complement

Note. For a block matrix $M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ with D invertible, the **Schur complement of D in M** is

$$S = A - BD^{-1}C.$$

Key identities (when the required inverses exist):

- Determinant: $\det(M) = \det(D)\det(S)$.

- Block inverse: $M^{-1} = \begin{bmatrix} S^{-1} & -S^{-1}BD^{-1} \\ -D^{-1}CS^{-1} & D^{-1} + D^{-1}CS^{-1}BD^{-1} \end{bmatrix}.$

- If M is symmetric positive (semi)definite and D is invertible, then S is also positive (semi)definite.

Symmetrically, if A is invertible, the Schur complement of A is $D - CA^{-1}B$. In Gaussian conditioning, $\mathrm{Cov}(a \mid b)$ equals the Schur complement of Σ_{bb} in the joint covariance.

2 Conceptual Foundation

2.1 What is Machine Learning?

"ML is a mathematization of epistemolagy!". In philosophy, it is the science of knowledge, the science of what can be known. This is relevant, because in ML we are interested in systems that produce/generate knowledge.

The goal is then to observe 'reality' and draw conclusions from the observations. This can be seen as a perception-action cycle. Where perception is the result of our observations (typically in a data space $\mathcal{X}$) and the actions are part of a hypothesis space. To go from the data space to the hypothesis class $\mathcal{C}$ we generally use an algorithm A. See Figure 1 for an overview.
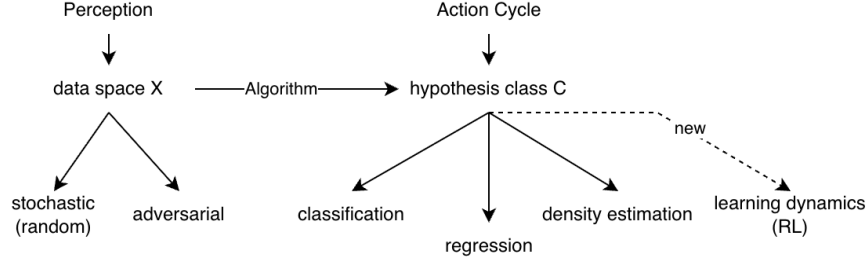
Figure 1: Overview ML

Information processing occurs when $|\mathcal{X}| \gg |\mathcal{C}|$. Taking the example of combinatorial optimization problems, $|\mathcal{X}|$ would be the space of weighted graphs, and we look for a color of the graph or something similar. Then $|\mathcal{X}| \approx K^{\binom{n}{2}}$ and $|\mathcal{C}| \approx e^{n \log n}$ so we obserge that $|\mathcal{X}|$ is much larger.

2.2 Conceptual foundation of inference

1. Perception of realits is mediated by data of senses/sensors

2. Data are stochastic $\rightarrow$ porbabilistic

3. Sensing restricts us to selected aspects of reality

4. Humans interpret data by a huge reduction in degrees of freedom
   $|x| \gg |e|$ (space of graphs $\gg$ space of colorings, cycles, spanning trees)

5. Tuple ({data }, {hypotheses }) define models:

$$\mathcal{A} : \mathcal{X} \longrightarrow \mathcal{C}$$
$$x \longmapsto c = \mathcal{A}(x)$$

6. Experiments ϵ provide us with data

7. Learning means interpreting X w.r.t hypotheses $\mathcal{C}$

2.3 Artificial Intelligence

Taking a high level view. We live in very high dimensional data, which we are not able to fully process. We use algorithms to make sense of the data, and this then informs our values, from this we get the following relation

$$\text{Data} \longrightarrow \text{Algorithms } \mathcal{A} \longrightarrow \text{Values}$$

In epistemology, we differentiate between **deduction** and **induction**. Deduction is a form of reasoning in which the conclusion follows necessarily from the premises, while induction tries to generalize, that is, the conclusion goes beyond the premises and generally probabilistic. We can construct a model in which deduction and induction form a **feedback loop**, not two isolated

methods. In simpler terms, on one side we try to formulate axioms from our observations (empirical data), and on the other side we then use these axioms and derive logical consequences from them. This is not anything new, and this cycle generally informs the model of "Theory, Experiment, Computation" in science. What has changed with ML/AI is that we're now in the era of nonparametric modeling.

2.4   Extracting Value from Data - What is the problem?

- Algorithms that process inputs with noise compute random variables as outputs!

- Algorithms should compute typical solutions!

- When do algorithms generalize over noise/model mismatch?

- How can algorithms autonomously improve performance?

2.5   What does Generalization mean?

- Out-of-sample risk

$$\theta^\star (X') \sim \mathbb{P}^{\mathcal{A}}(\theta \mid X) \in \arg\min_{\mathbb{P}(.|.)} \mathbb{E}_{X'}\mathbb{E}_{\theta|X'}\mathbb{E}_{X''|X'} R(\theta, X'')$$

where X' is the training data and X'' is the test data, so thiw is the risk where θ is conditioned on the training data (trained the model).

- Log loss of posterior (risks and probabilities are dependent!)

$$\theta^\star (X') \sim \mathbb{P}^{\mathcal{A}}(\theta \mid X) \in \arg\min_{\mathbb{P}(.|.)} \mathbb{E}_{X'}\mathbb{E}_{\theta|X'}\mathbb{E}_{X''|X'} \left(-\log \mathbb{P}(\theta \mid X'')\right)$$

$$\in \arg\min_{\mathbb{P}(.|.)} \mathbb{E}_{X'}\mathbb{E}_{\theta|X'}\mathbb{E}_{X''|X'} \left(\beta R(\theta, X'') + \log Z\right)$$

- Posterior agreement

$$\theta^\star (X') \sim \mathbb{P}^{\mathcal{A}}(\theta \mid X) \in \arg\min_{\mathbb{P}(.|.)} \mathbb{E}_{X'}\mathbb{E}_{X''|X'} \left(-\log \mathbb{E}_{\theta|X'}\mathbb{P}(\theta \mid X'')\right)$$

2.6   Conceptional Foundation of Inference

- Our perception of "our world" / reality is mediated by data of senses / sensors.

- Our data are influenced by chance.

- Creatures interpret selected aspects of reality by hypotheses to "survive and reproduce"

- Data and hypotheses define models to enable judgements, decisions and actions.

- AI / ML: Algorithms define relations of data and hypotheses, e.g., they select models !

3 Fundamentals of Machine Learning

Machine learning is fundamentally about inferring models from data. At its core lies Bayes' rule, which relates the posterior distribution (model given data) to the likelihood and prior:

$$\mathbb{P}(\text{model} \mid \text{data}) = \frac{\mathbb{P}(\text{ data } \mid \text{ model })\mathbb{P}(\text{ model })}{\mathbb{P}(\text{ data })}$$

In the **maximum likelihood (ML)** approach, we select the model that maximizes the likelihood of observing the data:

$$\widehat{\text{model}} \in \arg \max_{\text{model}} \mathbb{P}(\text{data} \mid \text{model})$$

Under regularity conditions, the ML estimator $\widehat{\text{model}}_n$ is consistent, asymptotically normal, and asymptotically efficient.

Consistency: A point estimator $\hat{\theta}_n$ of the parameter $\theta = \theta_0$ is consistent if it converges in probability to the true parameter:

$$\forall \varepsilon > 0, \mathbb{P}\left(\left|\hat{\theta}_n - \theta_0\right| > \varepsilon\right) \xrightarrow{n \to \infty} 0$$

More formally, using the ε-δ definition:

$$\forall \theta, \forall \varepsilon, \delta > 0, \exists n_0, \forall n > n_0, \mathbb{P}\left(\left|\hat{\theta}_n - \theta\right| < \varepsilon\right) > 1 - \delta$$

Efficiency: An estimator $\hat{\theta}_n$ is efficient if it achieves the minimum mean squared error among all estimators:

$$\hat{\theta}_n = \arg \min_{\hat{\theta}} \mathbb{E}\left[\left(\hat{\theta}_n - \theta_0\right)^2\right]$$

The question then arises: how precisely can we estimate θ given n samples? The Cramér-Rao bound provides a fundamental lower bound on the variance of any unbiased estimator.

3.1 Efficiency: Cramér-Rao Bound

Problem: What is the best achievable precision for parameter estimation?

Given a likelihood $p(y \mid \theta)$ for $\theta \in \Theta$ and data $y_1, \ldots, y_n \sim p(y \mid \theta = \theta_0)$, we ask: How precisely can we estimate $\theta = \theta_0$ given n samples?

For an estimator $\hat{\theta}(y_1, \ldots, y_n)$, we measure precision via the mean squared error:

$$\mathbb{E}_{y|\theta}\left[(\hat{\theta} - \theta)^2\right]$$

Key definitions:

- Score: $\Lambda = \frac{\partial}{\partial \theta} \log p(y \mid \theta) = \frac{\frac{\partial}{\partial \theta} p(y|\theta)}{p(y|\theta)}$

- Bias: $b_{\hat{\theta}} = \mathbb{E}_{y|\theta}\left[\hat{\theta}(y_1, \ldots, y_n)\right] - \theta$

Expected score: The score has zero mean.

$$\mathbb{E}_{y|\theta}[\Lambda] = \int p(y \mid \theta) \frac{\frac{\partial}{\partial \theta} p(y \mid \theta)}{p(y \mid \theta)} dy$$
$$= \frac{\partial}{\partial \theta} \int p(y \mid \theta) dy = \frac{\partial}{\partial \theta} 1 = 0$$

Score-estimator product:

$$\mathbb{E}_{y|\theta}[\Lambda \hat{\theta}] = \int p(y \mid \theta) \frac{\frac{\partial}{\partial \theta} p(y \mid \theta)}{p(y \mid \theta)} \hat{\theta} dy$$
$$= \frac{\partial}{\partial \theta}\left(\int p(y \mid \theta) \hat{\theta} dy\right)$$
$$= \frac{\partial}{\partial \theta}\left(\mathbb{E}_{y|\theta} \hat{\theta}\right) = \frac{\partial}{\partial \theta}\left(b_{\hat{\theta}} + \theta\right) = \frac{\partial}{\partial \theta} b_{\hat{\theta}} + 1$$

Cross-correlation:

$$\mathbb{E}_{y|\theta} \left[(\Lambda - \mathbb{E}\Lambda) \left(\hat{\theta} - \mathbb{E}\hat{\theta} \right) \right] = \mathbb{E}_{y|\theta}[\Lambda\hat{\theta}] - \mathbb{E}_{y|\theta}[\Lambda]\mathbb{E}\hat{\theta} = \mathbb{E}_{y|\theta}[\Lambda\hat{\theta}]$$

since $\mathbb{E}[\Lambda] = 0$.

Cauchy-Schwarz inequality: Applying Cauchy-Schwarz to the cross-correlation:

$$\left(\mathbb{E}_{y|\theta}[\Lambda(\hat{\theta} - \mathbb{E}\hat{\theta})] \right)^2 \leq \mathbb{E}_{y|\theta} \left[\Lambda^2 \right] \mathbb{E}_{y|\theta} \left[(\hat{\theta} - \mathbb{E}\hat{\theta})^2 \right]$$

Expanding the right-hand side:

$$\mathbb{E}_{y|\theta} \left[(\hat{\theta} - \mathbb{E}\hat{\theta})^2 \right] = \mathbb{E}_{y|\theta} \left[(\hat{\theta} - \theta + \theta - \mathbb{E}\hat{\theta})^2 \right]$$
$$= \mathbb{E}_{y|\theta} \left[(\hat{\theta} - \theta)^2 \right] - b_{\hat{\theta}}^2$$

Therefore:

$$\left(\frac{\partial}{\partial\theta} b_{\hat{\theta}} + 1 \right)^2 \leq \mathbb{E}_{y|\theta} \left[\Lambda^2 \right] \left(\mathbb{E}_{y|\theta} \left[(\hat{\theta} - \theta)^2 \right] - b_{\hat{\theta}}^2 \right)$$

Rearranging yields the **Cramér-Rao bound**:

$$\mathbb{E}_{y|\theta} \left[(\hat{\theta} - \theta)^2 \right] \geq \frac{\left(\frac{\partial}{\partial\theta} b_{\hat{\theta}} + 1 \right)^2}{\mathbb{E}_{y|\theta} \left[\Lambda^2 \right]} + b_{\hat{\theta}}^2$$

Fisher information: The expected squared score is called the Fisher information:

$$I(\theta) := \mathbb{E}_{y|\theta} \left[\Lambda^2 \right] = \int p(y \mid \theta) \left(\frac{\partial}{\partial\theta} \log p(y \mid \theta) \right)^2 dy$$

It measures how much information the data contains about the parameter θ. Higher Fisher information means we can estimate θ more precisely.

Remarks:

- For unbiased estimators ($b_{\hat{\theta}} = 0$), the bound simplifies to $\mathbb{E}[(\hat{\theta} - \theta)^2] \geq 1/I(\theta)$.

- The bound reveals a trade-off for biased estimators: reducing bias derivative $\frac{\partial}{\partial\theta} b_{\hat{\theta}}$ vs. reducing squared bias $b_{\hat{\theta}}^2$. Unbiased estimators are not always optimal!

Fisher information for n i.i.d. samples:

$$I^{(n)}(\theta) = \mathbb{E}_{y_1,\ldots,y_n|\theta} \left[\Lambda^2 \right]$$
$$= \mathbb{E} \left[\left(\frac{\partial}{\partial\theta} \log p(y_1, \ldots, y_n \mid \theta) \right)^2 \right]$$
$$= \mathbb{E} \left[\left(\sum_{i=1}^n \frac{\partial}{\partial\theta} \log p(y_i \mid \theta) \right)^2 \right] = \mathbb{E} \left[\left(\sum_{i=1}^n \Lambda_i \right)^2 \right]$$
$$= \sum_{i=1}^n \mathbb{E} \left[\Lambda_i^2 \right] + \sum_{i \neq j} \mathbb{E} \left[\Lambda_i \right] \mathbb{E} \left[\Lambda_j \right]$$
$$= \sum_{i=1}^n \mathbb{E} \left[\Lambda_i^2 \right] = nI(\theta)$$

where the cross-terms vanish because $\mathbb{E}[\Lambda_i] = 0$ and the samples are independent.

Key insight: The Fisher information of n i.i.d. random variables is n times the Fisher information of a single random variable. This shows that precision improves linearly with sample size.

4 Regression

4.1 Act 1: High-dimensional regression is unstable

We assume X and y are distributed according to a distribution p_* (i.e. $X, y \sim p_*$), where the output follows a noisy linear model:

$$y = f_*(x) + \varepsilon \quad \text{with } \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

Here f_* is the true (unknown) regression function and ε is additive Gaussian noise with variance σ^2. Our task is to estimate f_* from training data $D = \{(x_i, y_i)\}_{i=1}^n \sim p_*$.

The problem in this form is not tractable because the space of all possible functions is too large. We therefore restrict ourselves to linear functions:

$$f_*(x) = \beta^\top x$$

where $\beta \in \mathbb{R}^d$ is a parameter vector. Given the Gaussian noise assumption, each observation has likelihood $p(y_i | x_i, \beta) = \mathcal{N}(\beta^\top x_i, \sigma^2)$. We solve for β using Maximum Likelihood Estimation (MLE):

$$\begin{aligned}
\widehat{\beta} &= \arg\max_{\beta \in \mathbb{R}^d} p(D \mid \beta) \\
&= \arg\max_\beta \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \beta^\top x_i)^2}{2\sigma^2}\right) \\
&= \arg\min_\beta \sum_{i=1}^n (y_i - \beta^\top x_i)^2 \\
&= \arg\min_\beta \mathrm{MSE}(D, \beta)
\end{aligned}$$

Maximizing the log-likelihood is equivalent to minimizing the mean squared error (MSE). The closed-form solution depends on whether we have more features than samples or vice versa:

$$\begin{aligned}
\widehat{\beta} &= (X^\top X)^{-1} X^\top y \quad \text{(when } d < n, \text{ more samples than features)} \\
&= X^\top (XX^\top)^{-1} y \quad \text{(when } d > n, \text{ more features than samples)}
\end{aligned}$$

These are algebraically equivalent by the Woodbury matrix identity. The first formula is the standard *ordinary least squares (OLS)* estimator, where

$$X = \begin{bmatrix} -x_1- \\ \vdots \\ -x_n- \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

This estimator has some interesting properties. It is unbiased and, by the Gauss-Markov Theorem, it is the best linear unbiased estimator (BLUE), i.e. it attains the smallest variance among all linear unbiased estimators. Thus, from the formula

$$\text{error} = \text{bias}^2 + \text{variance} + \text{noise}$$

we find that this estimator is the one with the smallest error of all the unbiased estimators. Then why does no-one use this estimator? If we introduce a bit of bias, we can significantly reduce the variance.

To understand the instability, we analyze $\mathrm{Var}(\hat{\beta})$ using the singular value decomposition (SVD) $X = UDV^\top$, where $U \in \mathbb{R}^{n \times n}$ and $V \in \mathbb{R}^{d \times d}$ are orthogonal, and D is diagonal with singular values $D_{11} \geq D_{22} \geq \ldots \geq 0$. Plugging this into the OLS formula:

$$\hat{\beta} = (X^\top X)^{-1} X^\top y = (VD^\top U^\top UDV^\top)^{-1} VD^\top U^\top y = VD^{-1}U^\top y$$

Since $y = X\beta_* + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$, and we multiply y by the deterministic matrix $VD^{-1}U^\top$, the estimator $\hat{\beta}$ is also Gaussian. Its variance is:

$$\mathrm{Var}(\hat{\beta}) = \mathrm{Var}(VD^{-1}U^\top y) = VD^{-1}U^\top \mathrm{Var}(y) UD^{-1}V^\top = \sigma^2 VD^{-2}V^\top = \sigma^2 \sum_{i \leq r} \frac{1}{D_{ii}^2} V_i V_i^\top$$

where $r = \text{rank}(X)$ and V_i is the i-th column of V (the i-th right singular vector).

The problem: In high-dimensional data, features are often correlated (e.g., pixel intensities in images, gene expressions). This makes X close to low-rank, so several singular values D_{ii} are very small. The variance contributions $1/D_{ii}^2$ then explode for these directions, causing massive instability in $\hat{\beta}$ even though it remains unbiased. Small noise in y gets amplified enormously in directions with small singular values, leading to wild predictions on test data.

4.2 Act 2: Stability via Regularization

The solution to the variance blow-up is to introduce regularization, which adds a small amount of bias in exchange for a large reduction in variance. We can derive regularization naturally from a Bayesian perspective.

The typical process of **Bayesian inference** goes through the following stages:

1. Prior $\beta \sim \mathcal{N}(0, \tau^2 I)$ — we assume β is drawn from a Gaussian centered at zero with variance τ^2. This encodes our belief that coefficients should not be too large.

2. Likelihood $p(D|\beta) = \prod_i \mathcal{N}(y_i | \beta^\top x_i, \sigma^2)$ — same Gaussian noise model as before.

3. Posterior (via Bayes' rule) $p(\beta|D) \propto p(\beta)p(D|\beta) \propto \exp\left(-\frac{1}{2\sigma^2}\text{MSE}(D, \beta) - \frac{1}{2\tau^2}\|\beta\|^2\right)$

The posterior combines the likelihood (fit to data) with the prior (regularization). To derive the optimization objective, we use the fact that maximizing the posterior probability is equivalent to minimizing its negative logarithm. From Bayes' rule:

$$p(\beta|D) \propto p(\beta)p(D|\beta) = \mathcal{N}(0, \tau^2 I) \cdot \prod_i \mathcal{N}(y_i | \beta^\top x_i, \sigma^2)$$

Taking the negative log (up to constant terms):

$$-\log p(\beta|D) \propto \underbrace{-\log p(\beta)}_{-\frac{1}{2\tau^2}\|\beta\|^2 + \text{const}} + \underbrace{-\log p(D|\beta)}_{-\sum_i \log \mathcal{N}(y_i|\beta^\top x_i, \sigma^2)}$$

For Gaussian distributions, $-\log \mathcal{N}(y|\mu, \sigma^2) = \frac{(y-\mu)^2}{2\sigma^2} + \text{const}$, so:

$$-\log p(\beta|D) \propto \frac{1}{2\tau^2}\|\beta\|^2 + \frac{1}{2\sigma^2}\sum_i (y_i - \beta^\top x_i)^2$$

Minimizing this gives the MAP (maximum a posteriori) estimate:

$$\hat{\beta}_{\text{MAP}} = \arg\min_\beta \left[\frac{1}{2\sigma^2}\sum_i (y_i - \beta^\top x_i)^2 + \frac{1}{2\tau^2}\|\beta\|^2\right]$$

$$= \arg\min_\beta \left[\sum_i \left(y_i - \beta^\top x_i\right)^2 + \lambda\|\beta\|^2\right]$$

This is precisely **ridge regression** with regularization parameter $\lambda = \sigma^2/\tau^2$. The ℓ^2 penalty $\|\beta\|^2$ shrinks coefficients toward zero. If we instead use a Laplace prior $p(\beta) \propto \exp(-|\beta|/\tau)$ with heavier tails, we obtain **lasso regression** with an ℓ^1 penalty $\|\beta\|_1$, which promotes sparsity.

The prior variance τ^2 controls the bias-variance trade-off: small τ^2 (big λ) means strong regularization (more bias, less variance), while large τ^2 (small λ) recovers OLS (no bias, high variance). The MAP (maximum a posteriori) solution is:

$$\hat{\beta}_{\text{MAP}} = \left(X^\top X + \lambda I\right)^{-1} X^\top y \tag{1}$$

Derivation. Start from the MAP/ridge objective in matrix form

$$J(\beta) = \sum_i (y_i - x_i^\top \beta)^2 + \lambda \|\beta\|^2 = \|y - X\beta\|^2 + \lambda \beta^\top \beta.$$

Differentiate and set the gradient to zero:

$$\nabla_\beta J(\beta) = -2X^\top(y - X\beta) + 2\lambda\beta$$
$$= 2\,(X^\top X + \lambda I)\beta - 2X^\top y \,=\, 0.$$

Thus the normal equations are $(X^\top X + \lambda I)\,\hat{\beta} = X^\top y$. For $\lambda > 0$, the matrix $X^\top X + \lambda I$ is positive definite and hence invertible, which yields the closed form in (1).

Compare this to OLS: the regularization term λI is added to $X^\top X$ before inversion, preventing ill-conditioning. Using SVD again to analyze the variance:

$$\text{Var}(\hat{\beta}_{\text{MAP}}) = \sigma^2 \sum_{i\leq r} \frac{D_{ii}^2}{\left(D_{ii}^2 + \lambda\right)^2} V_i V_i^\top \tag{2}$$

The key is the **shrinkage factor** $\frac{D_{ii}^2}{(D_{ii}^2 + \sigma^2/\tau^2)^2}$. For large singular values $(D_{ii}^2 \gg \sigma^2/\tau^2)$, this is close to $1/D_{ii}^2$ (like OLS). For small singular values $(D_{ii}^2 \ll \sigma^2/\tau^2)$, the factor is approximately $\tau^4 D_{ii}^2/\sigma^4$, which decays much more slowly than $1/D_{ii}^2$. This prevents variance blow-up in the problematic low-variance directions, stabilizing the estimator at the cost of introducing bias (shrinking coefficients toward zero).

4.3 Act 3: Polynomial regression via kernels

Now we change our assumption for $f_*(x)$ to allow for nonlinear functions. We model f_* as a linear function in an infinite-dimensional feature space:

$$f_*(x) = \varphi(x)^\top \beta_*$$

where $\beta_* \in \mathbb{R}^\infty$ and $\varphi(x)$ maps each input to an infinite-dimensional polynomial feature representation:

$$\varphi(X) = K_x\left(\frac{x_1^{\alpha_1}...x_d^{\alpha_d}}{\sqrt{\alpha_1!\ldots\alpha_d!}}\right)_{\alpha\in\mathbb{N}^d}$$

This includes all polynomial terms of all degrees. The normalization by factorials ensures the inner product has a clean closed form.

Remarkably, the inner product of two infinite-dimensional feature vectors yields the radial basis function (RBF) kernel. For $x, x' \in \mathbb{R}^a$:

$$\varphi(x)^\top \varphi(x') = K_{RBF}(x, x')$$
$$= \exp\left(-\frac{1}{2}||x - x'||^2\right)$$

This follows from the Taylor expansion of the exponential function. The RBF kernel measures similarity: it is 1 when $x = x'$ and decays as points move apart.

We still want to minimize the MSE, but now in the infinite-dimensional feature space:

$$\hat{\beta} = \arg\min_{\beta\in\mathbb{R}^\infty} \frac{1}{n}\sum_{i\leq n}\left(y_i - \varphi(x_i)^\top\beta\right)^2$$
$$= \Phi^\top\left(\Phi\Phi^\top\right)^{-1}y$$

where

$$\Phi = \begin{bmatrix} \varphi(x)^\top \\ \varphi(x_2)^\top \\ \varphi(x_n)^\top \end{bmatrix} \in \mathbb{R}^{n\times\infty}$$

Despite β living in infinite dimensions, the representer theorem guarantees the solution lies in the span of the training features, so we can work with the $n \times n$ Gram matrix $\Phi\Phi^\top$ instead of the infinite-dimensional feature space directly.

To make a prediction at test point x_*, we compute:

$$\hat{y}_* = \varphi\left(x_*\right)^\top \hat{\beta}$$
$$= \varphi\left(x_*\right)^\top \Phi^\top \left(\Phi\Phi^\top\right)^{-1} y$$
$$= k(x_*)^\top K^{-1} y$$

where $k(x_*) = \left(\varphi\left(x_*\right)^\top \varphi\left(x_i\right)\right)_{1 \leq i \leq n} = (K_{RBF}(x_*, x_i))_{1 \leq i \leq n}$ is an n-dimensional vector of kernel evaluations between the test point and each training point, and $K_{ij} = \varphi(x_i)^\top \varphi(x_j) = K_{RBF}(x_i, x_j)$ is the $n \times n$ kernel matrix.

This is the **kernel trick**: we never explicitly construct the infinite-dimensional $\varphi(\cdot)$. Instead, we only compute inner products via the kernel function K_{RBF}, which can be evaluated in closed form. The prediction is a weighted combination of training outputs, where the weights depend on how similar the test point is to each training point.

The problem is that the inversion of the matrix is $O(n^3)$, which becomes costly for large datasets even though we avoided the infinite feature map explicitly.

4.4 Act 4: Neural Networks

We assume f_* has only a single, very wide hidden layer.

$$f_*(X) = \frac{1}{\sqrt{m}} \sum_{i \leq m} \alpha_i \phi\left(\omega_i^\top X\right)$$

where ϕ is a nonlinear activation function (e.g. ReLU, tanh), and the network has m hidden units. The parameters are $\theta = \{\alpha_i, w_i\}_{i \leq m}$, i.e. both the output weights α_i and the input weights w_i are learned. We initialize with

$$\theta_0 \sim \mathcal{N}\left(0, w^2\right)$$

and we update our parameters using gradient descent.

$$\theta_{t+1} \leftarrow \theta_t - \eta \nabla_\theta \operatorname{MSE}(D, \theta_t)$$

The gradient can be written in matrix form as

$$\nabla_\theta \operatorname{MSE}\left(D, \theta_t\right) = \widetilde{\Phi}_t^\top \left(f_t - y\right)$$

where

$$\tilde{\Phi}_t = \left(-\nabla_\theta f\left(x_i; \theta_t\right)^\top -\right)_{i \leq n} \in \mathbb{R}^{n \times |\theta|} \quad \text{and} \quad f_t = (f(x_i; \theta_t))_{i \leq n} \in \mathbb{R}^n$$

Here $\tilde{\Phi}_t$ is the feature matrix whose i-th row is the gradient of the network output with respect to all parameters, evaluated at data point x_i and current parameters θ_t.

In the *lazy training regime* (small learning rate, wide network), the parameters stay close to initialization, so we can linearize the network via a first-order Taylor expansion around θ_0:

$$f_t \approx f_0 + \tilde{\Phi}_0 \left(\theta_t - \theta_0\right)$$

Assuming the feature matrix $\tilde{\Phi}_t$ remains approximately constant at $\tilde{\Phi}_0$ (which holds when $m \to \infty$), gradient flow yields

$$\theta_t - \theta_0 = \tilde{\Phi}_0^\top \left(\tilde{\Phi}_0 \tilde{\Phi}_0^\top\right)^{-1} \left(f_t - f_0\right)$$

This says the parameter change lies in the span of the gradients and is chosen to optimally fit the training residuals.

Now let x_* be a test point. Plugging the linearization into the prediction yields

$$f_t\left(x_*\right) \approx f_0\left(x_*\right) + \nabla_\theta f\left(x_*, \theta_0\right)^\top \tilde{\Phi}_0^\top \left(\tilde{\Phi}_0 \tilde{\Phi}_0^\top\right)^{-1} \left(f_t - f_0\right)$$

Define the *neural tangent kernel (NTK) K* with entries

$$K_{ij} = \nabla_\theta f(x_i, \theta_0)^\top \nabla_\theta f(x_j, \theta_0) = \left[\tilde{\Phi}_0 \tilde{\Phi}_0^\top\right]_{ij}$$

and similarly $k(x_*) = \left(\nabla_\theta f(x_*, \theta_0)^\top \nabla_\theta f(x_i, \theta_0) \right)_{i \leq n}$.

In the infinite-width limit ($m \to \infty$), the random initialization ensures $f_0(x) \to 0$ for all x (the outputs average out), and after infinite training time ($t \to \infty$), gradient descent drives the training residual to zero so $f_t \to y$. The prediction becomes

$$f_\infty(x_*) = k(x_*)^\top K^{-1} y$$

This is exactly the result we obtained in Act 3 for kernel regression.

Conclusion: Gradient descent on a very wide neural network operates in a *kernel regime*, where training is equivalent to kernel ridge regression with the neural tangent kernel. The NTK is determined by the architecture and activation function, but the solution has the same closed-form structure $k(x_*)^\top K^{-1} y$ as any other kernel method. In practice, finite-width networks can escape this regime and learn richer, feature-learning representations—this lazy limit is a useful theoretical baseline.

4.5 Some Things from the Slides

MAP, ERM, and the conditional mean. For squared loss and any hypothesis class rich enough to contain the regression function, the Bayes-optimal solution is the conditional expectation $f^\star(x) = \mathbb{E}[Y \mid X = x]$, i.e.

$$f^\star \in \arg \min_f \mathbb{E}_{X,Y} \left[(Y - f(X))^2 \right].$$

In practice $P(Y \mid X)$ is unknown, so we either (i) postulate a parametric model and perform maximum likelihood / MAP (e.g., assume $Y \mid X \sim \mathcal{N}(f_\beta(X), \sigma^2)$ and maximize $\sum_i \log p(y_i \mid x_i, \beta)$) or (ii) minimize the empirical risk $\sum_i (y_i - f(x_i))^2$ directly. For well-behaved models these two routes coincide, which explains why classical ERM with squared loss reproduces the MAP estimator of a Gaussian noise model.

Gauss–Markov optimality. Ordinary least squares does not merely give *a* solution, it gives the best linear unbiased estimator (BLUE). Consider any linear estimator $\tilde\theta = c^\top y$ that is unbiased for $a^\top \beta$ (i.e., $\mathbb{E}[\tilde\theta] = a^\top \beta$). The Gauss–Markov theorem states

$$\mathrm{Var}(a^\top \hat\beta) \leq \mathrm{Var}(\tilde\theta),$$

where $\hat\beta = (X^\top X)^{-1} X^\top y$ is the OLS solution and a is an arbitrary vector. Intuitively, any other unbiased linear estimator differs from OLS by an additive linear operator D with $a^\top D X = 0$, which only inflates variance through the positive semi-definite term $DD^\top$. This reinforces the motivation for OLS (or its regularized cousins) when unbiasedness and linearity are desired.

Bias–variance decomposition. Suppose we highlight a specific input x and view the learned regressor $\hat f$ as a random variable (due to sampling various training sets). The expected squared prediction error decomposes into variance, bias, and irreducible noise:

$$\mathbb{E}_D \mathbb{E}_{Y|X=x} \left[(\hat f(x) - Y)^2 \right] = \underbrace{\mathbb{E}_D \left[(\hat f(x) - \mathbb{E}_D[\hat f(x)])^2 \right]}_{\text{variance}} + \underbrace{\left(\mathbb{E}_D[\hat f(x)] - \mathbb{E}[Y \mid X = x] \right)^2}_{\text{bias}^2} + \underbrace{\mathrm{Var}(Y \mid X = x)}_{\text{noise}}.$$

Low-capacity models (small hypothesis classes) produce high bias and low variance, while expressive models produce the opposite. Managing this trade-off is the crux of regularization and model selection.

Shrinkage beyond ridge and lasso. Ridge (ℓ_2) and lasso (ℓ_1) penalties are instances of a broader shrinkage family

$$\hat\beta = \arg \min_\beta \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^d x_{i,j} \beta_j \right)^2 + \lambda \sum_{j=1}^d |\beta_j|^q, \quad q \in (0, \infty].$$

Varying q changes the geometry of the constraint set: $q = 2$ yields spherical ridge contours, $q = 1$ produces diamond-shaped lasso constraints whose corners encourage sparsity, and $q < 1$

(non-convex) promotes even stronger sparsity at the cost of more difficult optimization. These shrinkage priors can be interpreted as MAP estimators with different prior distributions on β (Gaussian, Laplace, etc.) and help calibrate the bias–variance compromise by shrinking noisy coefficients toward zero. In practice we trace the coefficient paths as the tuning parameter (either λ or the effective degrees of freedom) varies, and select the desired amount of shrinkage via cross-validation on held-out data. Ridge paths shrink smoothly without hitting zero, whereas lasso paths *do* cross zero, enabling feature selection alongside regularization.

5   Representations

Machine learning algorithms only see the world through the representations we choose. Good representations clarify the objective (expected risk), guide the empirical procedures we rely on (empirical risk and test evaluation), and encode the structure of the data via adequate feature, scale, and mathematical spaces.

5.1   Expected vs. empirical risk

Given a hypothesis $f \in \mathcal{C}$, a loss function Q and random variables (X, Y), the conditional expected risk is

$$R(f, X) = \int Q(Y, f(X)) \, \mathbb{P}(Y \mid X) \, dY,$$

while the total expected risk integrates over the data distribution

$$R(f) = \int R(f, X) \, \mathbb{P}(X) \, dX = \iint Q(Y, f(X)) \, \mathbb{P}(X, Y) \, dX \, dY.$$

Learning is phrased as minimizing $R(f)$, but we only have data. With a training sample $Z_{\text{train}} = \{(X_i, Y_i)\}_{i=1}^{n}$, the empirical risk (training error) of an estimator $\hat{f}_n$ is

$$\widehat{R}(\hat{f}_n, Z_{\text{train}}) = \frac{1}{n} \sum_{i=1}^{n} Q(Y_i, \hat{f}_n(X_i)), \qquad \hat{f}_n \in \arg\min_{f \in \mathcal{C}} \widehat{R}(f, Z_{\text{train}}).$$

The test set $Z_{\text{test}} = \{(X_j, Y_j)\}_{j=n+1}^{n+m}$ yields an unbiased estimate of $R(\hat{f}_n)$ *only* if it is held out until the final estimator is fixed:

$$\widehat{R}(\hat{f}_n, Z_{\text{test}}) = \frac{1}{m} \sum_{j=n+1}^{n+m} Q(Y_j, \hat{f}_n(X_j)).$$

Whenever we adapt the estimator or its hyperparameters using the test set, we introduce dependencies between the model and the data and obtain a too optimistic risk estimate. Statistical learning therefore mandates the strict separation of training, validation, and testing.

5.2   Comparing algorithms on shared test data

To compare two learning algorithms $A^{(I)}$ and $A^{(II)}$ on the same dataset, we evaluate their paired losses on each test point j:

$$\Delta_j = Q(Y_j, \hat{f}_n^{(I)}(X_j)) - Q(Y_j, \hat{f}_n^{(II)}(X_j)), \qquad j = n+1, \ldots, n+m.$$

The sample mean $\overline{\Delta}$ and standard deviation $\text{std}(\Delta)$ of these paired differences quantify which model is better. If $\overline{\Delta} - 2 \, \text{std}(\Delta) > 0$, then algorithm $A^{(II)}$ is reliably superior to $A^{(I)}$. This "first compare, then average" rule mirrors paired t-tests and guards against spurious conclusions from aggregate metrics alone.

5.3   Data, feature, and measurement spaces

Representation begins with deciding *what* we measure:

- **Object space** $\mathcal{O}$**.** Objects (digits, patients, sounds) form the domain of discourse.

- **Measurements** X**.** A measurement maps tuples of objects into a codomain K: $X \colon \mathcal{O}^{(1)} \times \cdots \times \mathcal{O}^{(R)} \to K$. Measurements can be direct (pixel intensities) or derived (edges, mel-cepstral coefficients).

- **Feature space** $\mathcal{X}$**.** Selecting $\mathcal{X}$ fixes the admissible metric and invariances. Numeric $\mathbb{R}^d$, Boolean, or categorical spaces each encode different similarity notions.

Typical data types emerge from the arity of the measurement:

Monadic/vectorial: $X : \mathcal{O} \to \mathbb{R}^d$ for temperature maps or feature vectors.

Dyadic: $X : \mathcal{O}^{(1)} \times \mathcal{O}^{(2)} \to \mathbb{R}$ for user–item interactions or contingency tables.

Pairwise similarity: $X : \mathcal{O} \times \mathcal{O} \to \mathbb{R}$ for protein alignment scores or image patch similarities.

Polyadic/multiway: $X : \mathcal{O}^{(1)} \times \mathcal{O}^{(2)} \times \mathcal{O}^{(3)} \to \mathbb{R}$ such as person $\times$ behavior $\times$ trait tensors or preferential choice data.

The taxonomy clarifies whether the learner should expect absolute values, co-occurrence counts, or structured relational inputs.

5.4   Scale types and transformation invariances

Scale choices express which transformations should leave conclusions invariant:

- **Nominal scale:** only equality matters; any bijection preserves meaning.

- **Ordinal scale:** rankings are meaningful; order-preserving functions are admissible.

- **Interval scale:** information resides in differences; affine transformations $f(x) = ax + c$ with $a > 0$ are allowed (e.g., Fahrenheit).

- **Ratio scale:** differences and ratios matter; only scalings $f(x) = ax$, $a > 0$ preserve structure (e.g., Kelvin).

- **Absolute scale:** literal values matter; only the identity transformation is valid (e.g., exam grades).

Data whitening—scaling features by their standard deviation—is one practical way to enforce comparable dynamic ranges so that the chosen metric respects the intended invariances.

5.5   Mathematical spaces underlying representations

Different mathematical spaces formalize increasingly rich structures:

- **Topological spaces** $(X, \mathcal{J})$**:** $\mathcal{J}$ is a family of subsets containing X and $\varnothing$, closed under finite intersections and arbitrary unions. Topology captures neighborhood relations without prescribing distances.

- **Metric spaces** (X, d)**:** A metric d satisfies non-negativity, identity of indiscernibles, symmetry, and the triangle inequality (or the stronger ultrametric inequality). Metrics quantify distances and therefore induce topologies.

- **Euclidean vector spaces** (V, ϕ)**:** A vector space equipped with a scalar product ϕ satisfying distributivity, symmetry, homogeneity, and positive definiteness. The induced norm $\|x\| = \sqrt{\phi(x, x)}$ generalizes standard Euclidean geometry.

Every Euclidean space is metric and thus topological, but the converse need not hold. Representation design should match the amount of reliable structure (topological, metric, or vectorial) that the measurements truly support.

5.6   Probability spaces

Probability theory provides the formal language for risk:

- **Sample space** $\Omega = \{\omega_1, \ldots, \omega_N\}$ collects all elementary outcomes (e.g., sequences of coin flips).

- **Event algebra** $\mathcal{A} \subseteq 2^\Omega$ contains Ω and is closed under union, intersection, and set difference, so that compound statements about outcomes remain valid events.

- **Probability measure** $\mathbb{P} : \mathcal{A} \to [0, 1]$ assigns weights $p(\omega_i)$ to elementary events, satisfies $\mathbb{P}(\Omega) = 1$, and extends additively to events $A \in \mathcal{A}$ via $\mathbb{P}(A) = \sum_{\omega_i \in A} p(\omega_i)$.

The triple $(\Omega, \mathcal{A}, P)$ underpins expected risk: once we define the events (representations) and their probabilities, we can integrate losses and reason about generalization.

6 Gaussian Processes for Regression

Gaussian processes (GPs) turn Bayesian linear regression into a flexible, non-parametric model that reasons about functions via distributions over all possible outputs. The slides emphasize three pillars: (i) the Bayesian linear regression foundation, (ii) kernel engineering for encoding inductive biases, and (iii) prediction, validation, and application pipelines that exploit GP uncertainty.

6.1 From Bayesian linear regression to GPs

Linear regression with Gaussian noise assumes $Y = x^\top \beta + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. Placing a Gaussian prior on the weights, $\beta \sim \mathcal{N}(0, \Lambda^{-1})$, yields the posterior

$$p(\beta \mid X, y) = \mathcal{N}(\mu_\beta, \Sigma_\beta), \qquad \mu_\beta = (X^\top X + \sigma^2 \Lambda)^{-1} X^\top y, \quad \Sigma_\beta = \sigma^2 (X^\top X + \sigma^2 \Lambda)^{-1}.$$

Understanding the posterior: The posterior mean μ_β combines the data (via $X^\top X$ and $X^\top y$) with the prior precision Λ. This is exactly the ridge regression solution we saw in Chapter 4.2, where Λ acts as a regularization matrix (compare to equation 1). The term $X^\top X + \sigma^2 \Lambda$ plays the role of a regularized Hessian, stabilizing the inversion. The posterior covariance Σ_β (compare to equation 2) captures uncertainty about β: directions along which the data is uninformative (small eigenvalues of $X^\top X$) retain their prior uncertainty, while directions well-explained by data shrink toward zero. Note that all uncertainty decreases with sample size n (more rows in X), and the marginal posterior variance of the i-th weight is Σ_β^{ii}.

Observations are linear combinations of Gaussian variables, so the vector of outputs $y = X\beta + \varepsilon$ is jointly Gaussian with mean zero and covariance

$$\text{Cov}(y) = X\Lambda^{-1}X^\top + \sigma^2 I_n.$$

Derivation. Since $\beta \sim \mathcal{N}(0, \Lambda^{-1})$ and $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_n)$ are independent,

$$\mathbb{E}[y] = \mathbb{E}[X\beta + \varepsilon] = X\,\mathbb{E}[\beta] + \mathbb{E}[\varepsilon] = 0,$$
$$\text{Cov}(y) = \mathbb{E}\big[(y - \mathbb{E}y)(y - \mathbb{E}y)^\top\big] = \mathbb{E}\big[(X\beta + \varepsilon)(X\beta + \varepsilon)^\top\big]$$
$$= X\,\mathbb{E}[\beta\beta^\top]X^\top + \mathbb{E}[\varepsilon\varepsilon^\top] + X\,\mathbb{E}[\beta\varepsilon^\top] + \mathbb{E}[\varepsilon\beta^\top]X^\top$$
$$= X\,\Lambda^{-1}X^\top + \sigma^2 I_n,$$

where the cross terms vanish by independence (and zero means).

Defining $k(x_i, x_j) = x_i^\top \Lambda^{-1} x_j$ turns this covariance into a kernel (Gram) matrix K, so $y \sim \mathcal{N}(0, K + \sigma^2 I)$. Replacing the dot-product kernel by any valid positive semi-definite kernel function "kernelizes" Bayesian ridge regression; the resulting stochastic process over functions is a Gaussian process.

6.2 What is a Gaussian process?

A **Gaussian process (GP)** is a collection of random variables indexed by inputs (e.g., $x \in \mathbb{R}^d$) such that any finite subset has a joint Gaussian distribution. Equivalently, a GP is a *distribution over functions*. We write

$$f \sim \mathcal{GP}(m, k), \quad m(x) = \mathbb{E}[f(x)], \quad k(x, x') = \text{Cov}(f(x), f(x')).$$

For any inputs $X = [x_1, \ldots, x_n]$, the function values satisfy

$$f(X) = [f(x_1), \ldots, f(x_n)]^\top \sim \mathcal{N}(m(X), K), \quad K_{ij} = k(x_i, x_j).$$

With Gaussian observation noise $y = f(X) + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$, we obtain

$$y \sim \mathcal{N}(m(X), K + \sigma^2 I).$$

The mean function m and kernel k fully specify the prior over functions: the kernel encodes smoothness, invariances, and correlation structure. Conditioning the prior GP on data (X, y) yields a *posterior GP* with updated mean and covariance, whose predictive mean/variance coincide with the closed-form formulas presented below. This makes GPs a principled, non-parametric way to model functions with calibrated uncertainty.

6.3 Kernel design

Kernels encode similarity and thereby the structure of the functions we wish to learn:

- Valid kernels must be symmetric and generate positive semi-definite Gram matrices for any finite set of inputs. They implicitly act as inner products in (possibly infinite-dimensional) Hilbert spaces.

- Standard kernels on $\mathbb{R}^d$ include linear $k(x, x') = x^\top x'$, polynomial $k(x, x') = (x^\top x' + 1)^p$, squared exponential $k(x, x') = \sigma_f^2 \exp(-\|x - x'\|^2/(2\ell^2))$, rational quadratic, exponential, and periodic kernels. Each carries different invariances (smoothness, periodicity, etc.).

- Kernels compose: sums, products, positive scalings, or applying positive-coefficient polynomials / exponentials to a base kernel all preserve validity. This "kernel engineering" enables similarity measures on non-vector data such as strings, graphs (diffusion kernels), or probability distributions.

Designing an appropriate kernel is tantamount to choosing the hypothesis space for the GP.

6.4 Prediction with Gaussian processes

Given training data (X, y) and a test input $x_\star$, we are interested in $y_\star$. The joint distribution of y and $y_\star$ is Gaussian:

$$\begin{bmatrix} y \\ y_\star \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K + \sigma^2 I & k \\ k^\top & c \end{bmatrix}\right),$$

where $K = k(X, X)$, $k = k(x_\star, X)$ is the vector of cross-covariances, and $c = k(x_\star, x_\star) + \sigma^2$. This is a natural extension of the setup without $x_\star$. Conditioning a multivariate Gaussian gives the predictive distribution

$$p(y_\star \mid x_\star, X, y) = \mathcal{N}(\mu_\star, \sigma_\star^2), \quad \mu_\star = k^\top (K + \sigma^2 I)^{-1} y, \quad \sigma_\star^2 = c - k^\top (K + \sigma^2 I)^{-1} k.$$

Derivation. **Theorem (Conditional Gaussian).** Suppose

$$\begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}\right),$$

with $\mathbf{a}_1, \mathbf{u}_1 \in \mathbb{R}^e$, $\mathbf{a}_2, \mathbf{u}_2 \in \mathbb{R}^f$, and covariance blocks $\Sigma_{11} \in \mathbb{R}^{e \times e}$, $\Sigma_{12} \in \mathbb{R}^{e \times f}$, $\Sigma_{21} \in \mathbb{R}^{f \times e}$, $\Sigma_{22} \in \mathbb{R}^{f \times f}$ positive semidefinite. Then the conditional distribution of $\mathbf{a}_2$ given $\mathbf{a}_1 = \mathbf{z}$ is

$$p(\mathbf{a}_2 \mid \mathbf{a}_1 = \mathbf{z}) = \mathcal{N}\left(\mathbf{u}_2 + \Sigma_{21}\Sigma_{11}^{-1}(\mathbf{z} - \mathbf{u}_1), \ \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}\right).$$

Hence GPs output **both a mean prediction and an uncertainty quantification**. This is however not the real uncertainty about the function value at $x_\star$, but the uncertainty we get inside our model, which is constrained to gaussian processes with a specific kernel.

Algorithm Prediction with Gaussian Processes

Require: n observed data $(\mathbf{X} = (x_1, \ldots, x_n)^\top \in \mathbb{R}^{n \times d}, \mathbf{y} \in \mathbb{R}^n)$, kernel function k, noise variance σ^2, new data point $x_{n+1} \in \mathbb{R}^d$

$\quad \mathbf{K} \leftarrow (k(x_i, x_j))_{1 \le i,j \le n}$ // Compute kernel matrix

$\quad \mathbf{k} \leftarrow (k(x_{n+1}, x_i))_{1 \le i \le n}$ // Similarity of new data point and observed data

$\quad \mu_{y_{n+1}} \leftarrow \mathbf{k}^\top (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$ // Mean of predictive distribution

$\quad \sigma_{y_{n+1}}^2 \leftarrow k(x_{n+1}, x_{n+1}) - \mathbf{k}^\top (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{k}$ // Variance of predictive distribution

$\quad$ **return** $\mathcal{N}(y_{n+1} | \mu_{y_{n+1}}, \sigma_{y_{n+1}}^2)$ // Return predictive distribution

---

The prediction algorithm returns a distribution function. The prediction at x_{n+1} yields a mean value $\mu_{y_{n+1}}$ and a variance $\sigma_{y_{n+1}}^2$. Furthermore, samples y_{n+1} can be drawn from this distribution.

6.5 Validating kernels and hyperparameters

Practical GP performance depends on hyperparameters such as the length-scale ℓ, signal variance σ_f^2, or kernel choice. The lecture highlights:

- Evidence maximization (type-II ML) to optimize hyperparameters by maximizing $\log p(y \mid X, \theta)$.

- Cross-validation schemes (random splits, leave-one-out) that rank kernels by predictive performance on held-out data. Synthetic experiments show that proper scoring rules recover the data-generating kernel, while real data (e.g., power plant energy output) can exhibit different optima depending on the scoring metric (squared exponential vs. periodic kernels).

- Bayesian comparison of "teacher" and "student" kernels: evaluate how well a candidate kernel matches data generated from another kernel by comparing their posterior predictive distributions.

Kernel validation is therefore an empirical model-selection layer that complements theoretical kernel properties.

6.6 Applications: control and fMRI

Because GPs model functions with calibrated uncertainty, they are appealing for safety-critical and data-efficient applications:

- **Safe Bayesian optimization for control.** In automatic controller tuning (e.g., quadrotor flight), the unknown performance function over controller parameters is modeled as a GP. Safe optimization explores only parameter settings whose predicted performance exceeds a safety threshold with high probability, gradually expanding the safe set and converging to the global optimum using few evaluations.

- **Robust controller design.** GPs capture both the mean and variance of system responses, enabling controllers that respect safety constraints while improving performance across uncertain dynamics. Comparisons to hand-tuned controllers highlight faster convergence and higher reliability.

These case studies reinforce that GP regression is more than a curve fitting tool—it is a probabilistic modeling framework that unifies inference, kernel engineering, and safety-aware decision making.

7 Ensemble Methods

Ensemble methods combine multiple base learners to obtain a predictor whose bias, variance, or loss surface is superior to any constituent model. By averaging or sequentially correcting learners trained on diverse views of the data, ensembles produce more stable predictions, calibrated uncertainties, and richer inductive biases than single models. This chapter summarizes the bootstrap-based family (bagging and random forests), boosting, and more general stacking strategies, with an emphasis on how they target different points along the bias–variance trade-off.

7.1 Why ensembles help

Let $\hat{f}(x)$ be a single hypothesis trained on data D. Ensembles form $\hat{f}_{\text{ens}}(x) = \sum_{b=1}^{B} w_b \hat{f}_b(x)$ from base learners $\hat{f}_b$. Two canonical effects explain their success:

- **Variance reduction.** If the $\hat{f}_b$'s are identically distributed with variance σ^2 and pairwise correlation ρ, then

$$\text{Var}\big[\hat{f}_{\text{avg}}(x)\big] = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2,$$

  which shrinks as B increases whenever $\rho < 1$. Bagging manipulates the data (bootstrap samples) to drive correlations down and thus stabilize high-variance learners such as decision trees or neural networks trained on small data.

Derivation. For the average predictor $\hat{f}_{\text{avg}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}_b(x)$,

$$\text{Var}[\hat{f}(x)] = \mathbb{E}_D\left(\hat{f}(X) - \mathbb{E}_D\hat{f}(X)\right)^2$$

$$= \mathbb{E}_D\left(\frac{1}{B}\sum_{i=1}^{B}\hat{f}_i(x) - \frac{1}{B}\sum_{i=1}^{B}\mathbb{E}_D\hat{f}_i(x)\right)^2$$

$$= \mathbb{E}_D\left(\frac{1}{B}\sum_{i=1}^{B}\left(\hat{f}_i(x) - \mathbb{E}_D\hat{f}_i(x)\right)\right)^2$$

$$= \frac{1}{B^2}\sum_{i=1}^{B}\text{Var}_D\left[\hat{f}_i(x)\right] + \frac{1}{B^2}\sum_{i\neq j}\text{Cov}\left[\hat{f}_i(x), \hat{f}_j(x)\right]$$

If all base learners share variance σ^2 and pairwise covariance $\text{Cov}(\hat{f}_i, \hat{f}_j) = \rho\,\sigma^2$,

$$\text{Var}\big[\hat{f}_{\text{avg}}(x)\big] = \frac{B}{B^2}\sigma^2 + \frac{B(B-1)}{B^2}\,\rho\,\sigma^2$$

$$= \frac{1}{B}\,\sigma^2 + \left(1 - \frac{1}{B}\right)\rho\,\sigma^2$$

$$= \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2.$$

Thus variance decreases like $1/B$ when correlations are small ($\rho \approx 0$), while residual correlation limits the gain.

- **Bias correction.** Sequential ensembles such as boosting fit a series of weak learners to the residuals (negative gradients) of the current model. Each stage nudges the predictor toward lower bias and can transform a barely better-than-random base learner into a strong classifier.

These mechanisms are complementary: bagging primarily attacks variance, boosting primarily attacks bias, and stacking blends heterogeneous models to capture complementary inductive biases.

7.2 Boosting

For now we're only interested in classification.

Given a Dataset $D = \{(x_i, y_i)\}_{i=1} \subseteq \mathbb{R}^d \times \{-1, +1\} \sim p_*$, where y_i are labels and x_i are features, with an unknown distribution p_*, we want to learn a classifier $G : \mathbb{R}^d \to \{-1, +1\}$ that minimizes the expected 0–1 loss:

$$L(G) = \mathbb{E}_{(X,Y)\sim p_*}[\mathbb{I}\{G(X) \neq Y\}]$$

there are three problems with this formulation:

- The distribution p_* is unknown, so we cannot compute the expected loss directly.

- The 0–1 loss is not differentiable, making optimization difficult.

- The hypothesis space of all classifiers $G : \mathbb{R}^d \to \{-1, +1\}$ is too large to search over.

To address these issues, we make the following changes:

- We replace the expected loss with the empirical loss on the training data:

$$\hat{L}(G, D) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\{G(x_i) \neq y_i\}$$

- We replace the 0–1 loss with a surrogate loss function $\ell : \mathbb{R} \to \mathbb{R}_+$ that is differentiable and convex, such as the exponential loss $\ell(z) = e^{-yG(x)}$.

- We restrict our hypothesis space to a set of weak learners $\mathcal{H}^A$ (A for additive), with

$$\mathcal{H}^A = \left\{ G_m | G_m(x) = \sum_{i \leq m} \beta_i f_i(x), f_i \in \mathcal{H}, \beta_i \in \mathbb{R} \right\}$$

This won't work with linear models. But with weak learners (e.g., decision stumps), we can use boosting to iteratively build a strong classifier by adding weak learners that minimize the surrogate loss on the training data.

So then our objective becomes:

$$\min_{G \in \mathcal{H}^A} \frac{1}{n} \sum_{i \leq n} \mathcal{L}(y_i; G(x_i))$$

with

$$\mathcal{L}(y_i; G(x)) = \exp(-y_i G(x_i)) = \exp(-y_i G_{m-1}(x_i)) \exp(-y\beta_m f_m(x_i)) = w_i^{m-1} \mathcal{L}(y_i; \beta_m f_m(x_i))$$

because $G_m(x) = G_{m-1}(x) + \beta_m f_m(x)$. What did we do? We decomposed the loss at step m into the loss at step $m-1$ and the contribution of the new weak learner f_m with weight β_m. So now we have two things to optimize: the weak learner f_m and its weight β_m.

and we can rewrite the objective as:

$$\min_{G_{m-1} \in \mathcal{H}_{m-1}^A} \min_{f_m \in \mathcal{H}, \beta_m \in \mathbb{R}} \sum_{i \leq n} w_i^{m-1} \mathcal{L}(y_i; \beta_m f_m(x_i))$$

7.3 Forward Stagewise Additive Modeling (FSAM)

is a greedy approach to solve this optimization problem. At each step m, we fix the previous model G_{m-1} and optimize for the new weak learner f_m and its weight β_m:

$G_0(x) \leftarrow 0$, $w_i^0 \leftarrow 1/n$ for all i
if $t = 1$ to m **then**
$\quad \min_{f_m \in \mathcal{H}, \beta_m \in \mathbb{R}} \frac{1}{n} \sum_{i \leq n} w_i^{t-1} \mathcal{L}(y_i; \beta_t f_t(x_i))$
$\quad w_i^t \leftarrow w_i^{m-1} \exp(-y_i G_{t-i}(x_i))$
end if

Future classifieres try to correct the mistakes of past classifiers by focusing more on misclassified examples (increasing their weights). This way, the ensemble learns from its errors and improves over time.

7.4 Gradient Descent Boosting

Wantde $G_\theta(x)$ where $\theta \in \Theta$ are parameters. We can use gradient descent to minimize the empirical loss:

$\theta_0 = \$$
for do $t = 1$ to T
    $\theta_t \leftarrow \theta_{t-1} - \alpha_t \nabla_\theta \mathcal{L}(D; \theta_{t-1})$
end for

At then end
$$\theta_T = \sum_{t \leq T} \alpha_t \nabla_\theta \mathcal{L}(D; \theta_{t-1})$$

The combined algorithm looks like this:

$G_0(x) \leftarrow 0$
for $t = 1$ to m **do**
    $r_i^{t-1} \leftarrow - \left[\frac{\partial \mathcal{L}(y_i; G(x_i))}{\partial G(x_i)} \right]_{G(x) = G_{t-1}(x)}$ for all i
    $\min_{f_t \in \mathcal{H}, \beta_t \in \mathbb{R}} \sum_{i \leq n} (r_i^{t-1} - \beta_t f_t(x_i))^2$
    $G_t(x) \leftarrow G_{t-1}(x) + \beta_t f_t(x)$
end for

8 Convex Optimization

Convex optimization studies problems whose objective and feasible sets are convex, guaranteeing that every local optimum is also global. These structure-driven guarantees let us design algorithms with provable convergence, quantify sensitivity, and provide certificates of optimality via dual variables. This chapter outlines the key definitions, canonical problem classes, duality theory, and foundational algorithms emphasized in the lecture notes.

8.1 Convex sets, functions, and problems

Let $C \subseteq \mathbb{R}^d$ be a set. C is **convex** if for any $x, y \in C$ and $\theta \in [0, 1]$, the convex combination $\theta x + (1 - \theta)y$ lies in C. Typical convex sets are affine subspaces, halfspaces, Euclidean balls, ellipsoids, spectrahedra, and probability simplices.

A function $f : \mathbb{R}^d \to \mathbb{R}$ is **convex** if its domain is convex and for every x, y and $\theta \in [0, 1]$

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y).$$

When f is differentiable, convexity is equivalent to the first-order inequality

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x),$$

meaning the affine tangent is a global under-estimator. If f is twice differentiable, convexity is equivalent to $\nabla^2 f(x) \succeq 0$ for all x in the interior of the domain. **Strong convexity** with parameter $m > 0$ strengthens the inequality to

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{m}{2}\|y - x\|^2,$$

implying a unique minimizer and improved convergence rates.

A generic convex optimization problem reads

$$\begin{aligned}
\min_{x \in \mathbb{R}^d} \quad & f_0(x) \\
\text{s.t.} \quad & f_i(x) \leq 0, \quad i = 1, \ldots, m, \\
& Ax = b,
\end{aligned}$$

where each f_i is convex and the equality constraints define an affine set. The feasible set $\{x \mid f_i(x) \leq 0, Ax = b\}$ must be nonempty for the problem to be well-posed. Convexity ensures that any Karush–Kuhn–Tucker (KKT) point is globally optimal.

8.2 Standard convex programs in machine learning

Many estimators from earlier chapters fit this template:

- **Least squares / ridge regression:** $f(x) = \frac{1}{2}\|Ax - b\|_2^2 + \frac{\lambda}{2}\|x\|_2^2$ is convex quadratic; ridge adds strong convexity, yielding the closed form in equation (1).

- **Lasso:** minimize $\frac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_1$. The ℓ_1 norm promotes sparsity via a polyhedral penalty while keeping the problem convex. Soft-thresholding is the proximal operator driving coordinate descent.

- **Support Vector Machines (SVM):** hinge-loss objectives $\sum_i \max(0, 1 - y_i w^\top x_i)$ with ℓ_2 regularization define convex problems whose dual has sparse support vectors.

- **Logistic regression:** the negative log-likelihood $\sum_i \log(1 + \exp(-y_i w^\top x_i))$ is convex. Adding ℓ_1 or ℓ_2 penalties yields generalized linear models solvable via gradient or Newton methods.

- **Matrix completion:** minimizing $\sum_{(i,j) \in \Omega}(X_{ij} - M_{ij})^2 + \lambda\|X\|_*$ uses the nuclear norm (convex surrogate of rank) to recover low-rank matrices.

Specialized subfamilies often admit faster algorithms:

- **Linear programs (LP):** $f_0(x) = c^\top x$, $f_i(x) = a_i^\top x - b_i$.

- **Quadratic programs (QP):** quadratic objective with linear constraints.

- **Second-order cone programs (SOCP)** and **semidefinite programs (SDP)** capture norms and PSD constraints, respectively, and power robust control plus covariance fitting.

8.3 Lagrangian duality and KKT conditions

For

$$\min_x f_0(x) \quad \text{s.t.} \quad f_i(x) \leq 0, \ Ax = b,$$

introduce multiplier $\lambda \geq 0$ for inequalities and ν for equalities. The **Lagrangian**

$$\mathcal{L}(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \nu^\top(Ax - b)$$

lower-bounds the primal objective: $g(\lambda, \nu) = \inf_x \mathcal{L}(x, \lambda, \nu) \leq f_0(x)$ for all feasible x. The **dual problem** maximizes $g(\lambda, \nu)$ subject to $\lambda \geq 0$. Weak duality ($g \leq p^\star$) always holds, while **strong duality** ($g^\star = p^\star$) holds under mild constraint qualifications such as Slater's condition (strict feasibility). Dual variables often have sensitivity interpretations, e.g., the effect of tightening constraints.

The **KKT conditions** describe optimality when strong duality holds:

Primal feasibility: $f_i(x^\star) \leq 0$, $Ax^\star = b$.

Dual feasibility: $\lambda^\star \geq 0$.

Stationarity: $\nabla f_0(x^\star) + \sum_i \lambda_i^\star \nabla f_i(x^\star) + A^\top \nu^\star = 0$.

Complementary slackness: $\lambda_i^\star f_i(x^\star) = 0$.

For unconstrained problems these reduce to $\nabla f_0(x^\star) = 0$, consistent with calculus. In SVMs, for example, KKT implies that only points on the margin have non-zero dual variables, explaining sparsity in the dual solution.

8.4 First-order algorithms

Most large-scale machine learning problems rely on first-order (gradient-based) methods with low per-iteration cost.

- **Gradient descent (GD).** Iterate $x_{k+1} = x_k - \eta_k \nabla f(x_k)$. For L-smooth convex f, constant step-size $\eta = 1/L$ yields $f(x_k) - f^\star \leq \frac{L\|x_0 - x^\star\|^2}{2k}$. If f is additionally m-strongly convex, the error contracts linearly: $f(x_k) - f^\star \leq \left(1 - \frac{m}{L}\right)^k (f(x_0) - f^\star)$.

- **Projected gradient.** For constrained problems with easy projections Π_C, update $x_{k+1} = \Pi_C(x_k - \eta \nabla f(x_k))$. This applies to simplex constraints, ℓ_2-balls, or PSD cones (via eigenvalue thresholding).

- **Subgradient methods.** When f is convex but nonsmooth (e.g., hinge or ℓ_1 norms), use $x_{k+1} = x_k - \eta g_k$ with $g_k \in \partial f(x_k)$. Convergence is $O(1/\sqrt{k})$ for general convex functions.

- **Accelerated methods.** Nesterov's accelerated gradient adds a momentum term achieving $f(x_k) - f^\star = O(1/k^2)$ for L-smooth convex objectives, approaching optimal complexity for first-order methods.

- **Stochastic gradients.** SGD and its variants (momentum, Adam, Adagrad) replace ∇f with unbiased estimates, giving scalable training on massive datasets. Convexity enables convergence proofs, e.g., $\mathbb{E}[f(\bar{x}_k)] - f^\star = O(1/\sqrt{k})$ with diminishing steps.

8.5 Second-order and interior-point methods

Second-order methods leverage curvature for faster local convergence.

- **Newton's method.** Update $x_{k+1} = x_k - \nabla^2 f(x_k)^{-1} \nabla f(x_k)$. Under mild conditions, Newton steps enjoy quadratic convergence near $x^\star$, but each iteration requires solving a linear system (often via Cholesky or conjugate gradients). Damped Newton with line search ensures global convergence for convex f.

- **Quasi-Newton methods.** BFGS or L-BFGS approximate Hessians by low-rank updates, offering superlinear convergence without storing the full Hessian. Widely used for logistic regression and other smooth convex losses.

- **Interior-point methods (IPM).** For constrained problems, IPMs solve a sequence of barrier subproblems

$$\phi_\mu(x) = f_0(x) - \mu \sum_{i=1}^{m} \log(-f_i(x))$$

using Newton steps. As $\mu \to 0$, solutions trace the *central path* toward the primal optimum while satisfying constraints. IPMs bring polynomial-time worst-case guarantees for LPs, QPs, SOCPs, and SDPs, albeit with higher per-iteration complexity than first-order methods.

8.6 Proximal, splitting, and decomposition techniques

When $f = g + h$ with g smooth and h simple but possibly nonsmooth, **proximal methods** exploit the proximal operator

$$\text{prox}_{\eta h}(v) = \arg\min_x \left\{ h(x) + \frac{1}{2\eta} \|x - v\|^2 \right\}.$$

Common examples include soft-thresholding (for ℓ_1 norms) or projection onto convex sets. Algorithms include:

- **Proximal gradient / ISTA**: $x_{k+1} = \text{prox}_{\eta h}(x_k - \eta \nabla g(x_k))$. FISTA accelerates this to $O(1/k^2)$ convergence.

- **Alternating direction method of multipliers (ADMM)**: solves $\min_x g(x) + h(z)$ subject to $Mx + Nz = c$ by alternating minimization with dual updates. ADMM decomposes problems across blocks, enabling distributed or parallel implementations (e.g., consensus optimization).

- **Augmented Lagrangian methods**: strengthen dual ascent by penalizing constraint violation quadratically, improving stability compared to vanilla dual methods.

8.7 Sensitivity, certificates, and modeling tips

- **Dual variables as sensitivities.** At optimum, $\lambda_i^\star$ equals the marginal increase in the optimal value if constraint i is tightened. Inspecting $\lambda^\star$ highlights bottleneck constraints or active data points.

- **Certificates of optimality.** A feasible primal-dual pair satisfying KKT provides a certificate. Numerically, interior-point solvers report the primal-dual gap $\|f(x) - g(\lambda, \nu)\|$, which bounds suboptimality.

- **Scaling.** Normalize features and constraints so that Hessians and gradients are well-conditioned; this accelerates both first- and second-order methods.

- **Modeling systems.** Packages like CVX, CVXPY, or disciplined convex programming (DCP) frameworks enforce convexity by construction, automatically converting high-level specifications into standard cone programs.

- **Stopping criteria.** Monitor residuals, objective gaps, and constraint violation simultaneously; mere gradient norm may be insufficient when constraints are present.

Convex optimization thus supplies the theoretical backbone for many machine learning estimators and algorithms: it furnishes tractable objectives, certifiable solutions, and scalable methods tailored to problem structure. Understanding duality and algorithmic trade-offs lets practitioners choose the right solver for each model in the AML toolbox.

9 From Convex Optimization to SVM

For support vector machines (SVMs), we're interested in maximising the margin between classes. So given a dataset $D = \{(x_i, y_i)\}$ with $y_i \in \{-1, +1\}$ and guaranteed linear separability, we want to find a hyperplane defined by (w, w_0) such that $w^T x_i + w_0 > 0$ if $y_i = 1$ and $w^T x_i + w_0 < 0$ if $y_i = -1$. Or in other terms, we want to satisfy the constraints $y_i(w^T x_i + w_0) > 0$ for all i. The margin is defined as the distance from the hyperplane to the closest data point, which can be expressed as $\frac{2}{\|w\|}$.

Derivation. Given the hyperplane $\{x \mid w^\top x + w_0 = 0\}$, the signed distance of any point x to the hyperplane is

$$\text{dist}(x, \mathcal{H}) = \frac{w^\top x + w_0}{\|w\|}.$$

Because $y_i \in \{-1, +1\}$, we can enforce the scale of (w, w_0) by requiring that the closest points satisfy $y_i(w^\top x_i + w_0) = 1$. Under this normalization, there exist parallel "margin" hyperplanes

$$w^\top x + w_0 = 1 \quad \text{and} \quad w^\top x + w_0 = -1$$

touching the positive and negative classes, respectively. The perpendicular distance between these two planes is the geometric margin:

$$\gamma = \frac{1 - (-1)}{\|w\|} = \frac{2}{\|w\|}.$$

Maximizing γ therefore amounts to minimizing $\|w\|$ (or equivalently $\frac{1}{2}\|w\|^2$ for convenience) under the constraints $y_i(w^\top x_i + w_0) \geq 1$ for all i. This yields the hard-margin SVM formulation

$$\min_{w, w_0} \frac{1}{2}\|w\|^2 \quad \text{s.t.} \quad y_i(w^\top x_i + w_0) \geq 1, \ i = 1, \ldots, n,$$

whose solution maximizes the separating margin.

9.1 Slater's Condition

By assumption that the data is linearly separable, there exists a feasible point (w, w_0) satisfying $y_i(w^\top x_i + w_0) > 1$ for all i. This strictly feasible point ensures that Slater's condition holds, guaranteeing strong duality between the primal and dual SVM problems.

9.2 The Dual

Writing the hard-margin primal in standard form,

$$\min_{w, w_0} \quad \frac{1}{2}\|w\|^2$$
$$\text{s.t.} \quad y_i(w^\top x_i + w_0) - 1 \geq 0, \quad i = 1, \ldots, n,$$

introduce Lagrange multipliers $\alpha_i \geq 0$ for each margin constraint. The Lagrangian is

$$\mathcal{L}(w, w_0, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i \big(y_i(w^\top x_i + w_0) - 1\big).$$

To form the dual we minimize $\mathcal{L}$ over the primal variables. Setting derivatives to zero yields the stationarity conditions

$$\nabla_w \mathcal{L} = w - \sum_i \alpha_i y_i x_i = 0 \quad \Rightarrow \quad w = \sum_i \alpha_i y_i x_i,$$

$$\frac{\partial \mathcal{L}}{\partial w_0} = -\sum_i \alpha_i y_i = 0.$$

Substituting back gives the dual objective

$$g(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^\top x_j = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} ||w(\alpha)||^2,$$

with constraints $\alpha_i \geq 0$ and $\sum_i \alpha_i y_i = 0$. Hence the dual problem is the quadratic program

$$\max_{\alpha \in \mathbb{R}^n} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} ||w(\alpha)||^2$$

$$\text{s.t.} \quad \alpha_i \geq 0, \quad i = 1, \ldots, n,$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0.$$

This dual depends only on inner products $x_i^\top x_j$, enabling the kernel trick by replacing them with $k(x_i, x_j)$. The optimal weights follow from the KKT conditions: only training points with $\alpha_i^\star > 0$ (the *support vectors*) contribute to $w^\star = \sum_i \alpha_i^\star y_i x_i$. Complementary slackness enforces $y_i(w^{\star\top} x_i + w_0^\star) = 1$ for active support vectors, which can be used to recover $w_0^\star$ by averaging over any $\alpha_i^\star > 0$.

Note. The dual formulation is preferable when n (number of samples) is smaller than d (feature dimension) or when kernels project data into high-dimensional spaces. It also highlights that margin maximization depends only on a sparse subset of training examples.

10 PCA and GMMs for Anomaly Detection

The task of anomaly detection involves identifying data points that deviate significantly from the norm. In situations like banks needing to detect fraudulent transactions, or hospitals aiming to identify unusual patient health metrics, effective anomaly detection methods are crucial.

10.1 Anomalies

Typically, we have a space of all possible events $\mathcal{X}$ and a subset called the normal set $\mathcal{N} \subseteq \mathcal{X}$. An anomaly is any event $x \in \mathcal{X}$ such that $x \notin \mathcal{N}$. What makes this challenging is that the normal set $\mathcal{N}$ is often not explicitly defined, and we may only have access to a limited set of examples from $\mathcal{N}$. The way we approach this problem is to do a dimensionality reduction Π on the data to find a lower-dimensional representation that captures the essential structure of the normal data. We can then use this representation to identify anomalies.

The generalized approach looks like this:

1. An anomaly is an unlikely event.

2. We fit a model of a parametric family of distributions $\mathcal{H} = \{p(x; \theta) : \theta \in \Theta\}$

3. We define an anomaly score $-\log p_{\hat{\theta}}(x)$

We choose a GMM (Gaussian Mixture Model) as our parametric family of distributions. A GMM is a weighted sum of multiple Gaussian distributions, which allows us to model complex data distributions effectively. The parameters θ of the GMM include the means, covariances, and mixture weights of the individual Gaussian components. This is a good choice, because it has been observed that linear projections of high-dimensional distributions onto low-dimensional spaces resemble Gaussian distributions, thie can partially be explained by the Central Limit Theorem.

This leads us to the following algorithm for anomaly detection using PCA and GMMs:

1. Project the data linearly onto a lower-dimensional space (using PCA)

2. Fit a GMM to the projected data

10.2 Dimensionality Reduction

Given $X = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^D$, find a linear projection $\pi : \mathbb{R}^D \to \mathbb{R}^d$, with $d \ll D$, and such that $\pi(X)$ has a "sufficiently large" variance. So we want a projection π that maximizes the variance of the projected data points.

10.2.1 First Stage

We start with a very tractable case which might be a bit too simple. We look for a one-dimensional projection $\pi(x) = w^T x$, where $w \in \mathbb{R}^D$ is a unit vector (i.e., $\|w\|_2 = 1$). The variance of the projected data points is given by:

$$\text{Var}(\pi(X)) = \frac{1}{n} \sum_{i=1}^{n} (w^T x_i - w^T \mu)^2 = w^T \Sigma w \tag{3}$$

where $\mu = \frac{1}{n} \sum_{i=1}^{n} x_i$ is the mean of the data points, and $\Sigma = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)(x_i - \mu)^T$ is the covariance matrix of the data points. To find the optimal projection vector w, we need to solve the following optimization problem:

$$\max_{w \in \mathbb{R}^D} w^T \Sigma w \quad \text{subject to} \quad \|w\|_2 = 1 \tag{4}$$

Where the solution is given by the eigenvector of Σ corresponding to the largest eigenvalue.

10.2.2 General Case

To generalize this to a d-dimensional projection, we can define the projection as $\pi(x) = W^T x$, where $W \in \mathbb{R}^{D \times d}$ is a matrix with orthonormal columns (i.e., $W^T W = I_d$). We proceed similarly to the one-dimensional case, and we want to maximize the variance of the projected data points:

$$\text{Var}(\pi(X)) = \frac{1}{n} \sum_{i=1}^{n} \|W^T x_i - W^T \mu\|_2^2 = \text{tr}(W^T \Sigma W) \tag{5}$$

To find the optimal projection matrix W, we need to solve the following optimization problem:

$$\max_{W \in \mathbb{R}^{D \times d}} \text{tr}(W^T \Sigma W) \quad \text{subject to} \quad W^T W = I_d \tag{6}$$

The solution is given by the matrix W whose columns are the eigenvectors of Σ corresponding to the d largest eigenvalues.

10.3 Fitting a GMM

10.3.1 GMM Definition

A Gaussian Mixture Model (GMM) is a probabilistic model that assumes that the data is generated from a mixture of several Gaussian distributions. The probability density function of a GMM with K components is given by:

$$p(x; \theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x; \mu_k, \Sigma_k) \tag{7}$$

where π_k are the mixture weights (with $\sum_{k=1}^{K} \pi_k = 1$ and $\pi_k \geq 0$), μ_k are the means, and Σ_k are the covariance matrices of the individual Gaussian components. The parameters of the GMM are denoted by $\theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K}$.

10.4 Fitting

We again use a MLE approach, that is, we want to find the parameters $\hat{\theta}$ that maximize the likelihood of the observed data:

$$\log p_\theta(x) = \log \prod_{i=1}^{n} p(x_i; \theta) \tag{8}$$

$$= \sum_{i=1}^{n} \log p(x_i; \theta) \tag{9}$$

$$= \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i | \mu_k, \Sigma_k) \right) \tag{10}$$

However, this last term is difficult to optimize directly due to the presence of the logarithm of a sum. We make an interesting observation: if we knew which component generated each data point, the optimization would be much simpler. This leads us to introduce latent variables z_i that indicate the component responsible for generating each data point x_i. Specifically, we define z_i as a one-hot encoded vector where $z_{ik} = 1$ if the i-th data point was generated by the k-th component, and 0 otherwise. With these latent variables, we can express the complete-data log-likelihood as:

$$\log p_\theta(X, Z) = \log p_\theta(Z) p_\theta(X \mid Z) \tag{11}$$

$$= \log p_\theta(Z) + \log p_\theta(X \mid Z) \tag{12}$$

$$= \sum_{i \leq n} \log \pi_{Z_i} + \sum_{i \leq n} \log p_\theta(X_i \mid Z_i) \tag{13}$$

Where $Z = \{z_1, \ldots, z_n\}$ is the set of latent variables for all data points. This is way easier to optimize.

Now we have the setup, that we have $\log p_\theta(x)$ but we would like to work with $\log p_\theta(X, Z)$. We know that:

$$\log p_\theta(x) = \log \frac{p_\theta(x, z)}{p_\theta(z|x)} \tag{14}$$

$$\Rightarrow \mathbb{E}_{Z \sim q}[\log p_\theta(x)] = \mathbb{E}_{Z \sim q}\left[\log \frac{p_\theta(x, Z)}{p_\theta(Z|x)}\right] \tag{15}$$

$$\tag{16}$$

Since the left-hand side does not depend on Z, we can write:

$$\log p_\theta(x) = \mathbb{E}_{Z \sim q}\left[\log \frac{p_\theta(x, Z)}{p_\theta(Z|x)}\right] \tag{17}$$

$$= \mathbb{E}_{Z \sim q}\left[\log \frac{p_\theta(x, Z)}{p_\theta(Z|x)} \cdot \frac{q(Z)}{q(Z)}\right] \tag{18}$$

$$= \mathbb{E}_{Z \sim q}[\log p_\theta(x, Z) - \log q(Z)] + \mathbb{E}_{Z \sim q}\left[\log \frac{q(Z)}{p_\theta(Z|x)}\right] \tag{19}$$

$$= M(q, \theta) + \mathrm{KL}(q\|p_\theta(Z|x)) \tag{20}$$

Where we defined:

$$M(q, \theta) = \mathbb{E}_{Z \sim q}[\log p_\theta(x, Z) - \log q(Z)] \tag{21}$$

This gives us a lower bound on the log-likelihood, since the KL divergence is always non-negative:

$$\log p_\theta(x) \geq M(q, \theta) \tag{22}$$

and we have equality if and only if $q(Z) = p_\theta(Z|x)$.

10.5 EM Algorithm

The Expectation-Maximization (EM) algorithm is an iterative method used to find maximum likelihood estimates of parameters in statistical models with latent variables. In the context of fitting a GMM, the EM algorithm alternates between two main steps: the Expectation (E) step and the Maximization (M) step.

- **E-step:** In this step, we compute the expected value of the complete-data log-likelihood with respect to the current estimate of the parameters $\theta^{(t)}$. This involves calculating the posterior probabilities of the latent variables given the observed data and the current parameter estimates. Specifically, we set:

$$q^{(t+1)}(Z) = p_{\theta^{(t)}}(Z|X) \tag{23}$$

- **M-step:** In this step, we maximize the expected complete-data log-likelihood computed in the E-step with respect to the parameters θ. This gives us updated parameter estimates:

$$\theta^{(t+1)} = \arg \max_\theta M(q^{(t+1)}, \theta) \tag{24}$$

The EM algorithm iterates between these two steps until convergence, which is typically determined by checking if the change in the log-likelihood or the parameter estimates falls below a predefined threshold.

10.6 Validation Metrics

We want a function $\phi : \mathbb{R}^D \to \{0, 1\}$ that that classifies points as normal (0) or anomalous (1). Given a threshold τ. We have two objectives: (let $C = \{x : \phi(x) = 1\}$ be the set of points classified as anomalous, and A be the set of true anomalies)

- If $x \in A$, then $\phi(x) = 1$

$$\mathrm{Recall} = \frac{|C \cap A|}{|A|} \tag{25}$$

'Reatio of the correctly identified anomalous among **all truly anomalous** points'

How many of the actual anomalous points did we identify?

- If $\phi(x) = 1$, then $x \in A$ (Precision)

$$\text{Precision} = \frac{|C \cap A|}{|C|} \tag{26}$$

'Ratio of the correctly identified anomalous points among **all identified anomalous points**'

How many of the points we classified as anomalous are actually anomalous?

We can combine these two metrics into a single metric called the F1-score, which is the harmonic mean of precision and recall:

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} \tag{27}$$

The F1-score provides a balanced measure of the model's performance in identifying anomalies, taking into account both precision and recall.

10.7 Combined Pipeline

Given a set $X \in \mathbb{R}^D$ of "normal" points, we train an anomaly detector as follows.

1. We compute a projector $\pi : \mathbb{R}^D \to \mathbb{R}^d$ using PCA.

2. We then fit a pdf $p_\theta(\cdot)$ with k -components to $\{\pi(x) : x \in X\}$, using the EM algorithm.

3. For a new point, its "anomaly score" is $-\log p_\theta(\pi(x))$.

11 Active Learning

Labels are often far more expensive than unannotated data. Diagnosing rare diseases, auditing corporate fraud, or curating safety-critical driving scenarios all require expert feedback, yet only a small subset of examples truly shapes the model. Active learning asks how to collect a small but representative labeled sample under a strict budget by interactively choosing which points to query. This chapter develops three progressively richer settings:

1. A transductive problem where we explicitly distinguish between the domain we can sample from and the target region where we aim to be accurate.

2. A safety-critical optimization problem in which only safe queries are admissible.

3. A batch selection task where labeling decisions are taken in groups using a coverage heuristic.

We follow the same storyline: first define the information-theoretic principle, then demonstrate how it carries over to safe Bayesian optimization, and finally study the batch coverage formulation.

11.1 Transductive information gain

We start with the transductive view. The learner is handed a domain $\mathcal{X}$, a *target* subset $\mathcal{A} \subseteq \mathcal{X}$ on which performance is ultimately measured, and a sample space $\mathcal{S} \subseteq \mathcal{X}$ that can actually be queried. Let f be an unknown stochastic process indexed by $\mathcal{X}$. Observation noise is modeled via $y_x = f_x + \epsilon_x$, where ϵ_x is independent, mean-zero noise. At iteration n we already collected

$$\mathcal{D}_{n-1} = \{(x_i, y_i)\}_{i<n} \subseteq \mathcal{S} \times \mathbb{R},$$

and the goal is to select $x_n \in \mathcal{S}$ such that the new datum (x_n, y_{x_n}) maximally informs us about f over the target domain $\mathcal{A}$. This raises two central questions:

- How do we quantify the information gain that a particular point provides?

- Given this score, how do we efficiently pick $x_n \in \mathcal{S}$ to maximize it?

11.2 Information-based transductive learning

Information-based transductive learning (ITL) answers both questions by selecting the point that maximizes the conditional mutual information between the new observation and the restriction of f to $\mathcal{A}$:

$$x_n = \arg\max_{x \in \mathcal{S}} I\left(\{f_x\}_{x \in \mathcal{A}} ; y_x \mid \mathcal{D}_{n-1}\right)$$

When $f \sim \mathcal{GP}(\mu, k)$ is a Gaussian process with known mean and kernel, the mutual information admits a closed form,

$$I\left(\{f_x\}_{x \in \mathcal{A}} ; y_x \mid \mathcal{D}_{n-1}\right) = \frac{1}{2} \log \left(\frac{\mathrm{Var}\left(y_x \mid \mathcal{D}_{n-1}\right)}{\mathrm{Var}\left(y_x \mid \{f_x\}_{x \in \mathcal{A}}, \mathcal{D}_{n-1}\right)} \right).$$

The numerator is simply the predictive variance from the GP posterior—how uncertain we currently are about y_x before revealing its label. The denominator measures what uncertainty would remain after conditioning on the entire target restriction $\{f_x\}_{x \in \mathcal{A}}$; it shrinks when labeling x clarifies many points in $\mathcal{A}$. Therefore a large ratio pinpoints samples whose individual labels resolve substantial ambiguity precisely where performance matters. The logarithm turns this ratio into an additive gain, so selecting the point with the largest mutual information implements a principled exploration strategy that prioritizes variance reduction on $\mathcal{A}$ while respecting that we may only query within the feasible set $\mathcal{S}$.

This completes the transductive core of the chapter: we now possess an acquisition functional that explicitly connects feasible sampling locations to a target region of interest. The following sections show how the very same ingredients—separate sample and target domains, plus an information-theoretic score—power safety-critical optimization and batch selection.

11.3 Safe Bayesian optimization

We next apply ITL to a safety-critical setting: we no longer merely want to reduce uncertainty on $\mathcal{A}$, we must also ensure that every experimental design point is safe. This leads to the safe Bayesian optimization problem of maximizing an unknown stochastic process $f^\star$ over $\mathcal{X}$:

$$x^\star = \arg\max_{x \in \mathcal{X}} \mathbb{E}[f^\star(x)],$$

subject to the *safe set* $\mathcal{S}^\star = \{x \in \mathcal{X} : g^\star(x) \geq 0\}$, where $g^\star$ is another stochastic process describing safety. Iteratively we gather observations $y_i = f^\star(x_i)$ and $z_i = g^\star(x_i)$ for $x_i \in \mathcal{X}$, but future samples must not violate the unknown constraint $g^\star(x) \geq 0$. The challenge is therefore to pick x_n that improves our estimate of the maximum value while remaining in the safe region with high probability.

11.4 Safe Bayesian optimization via ITL

To enforce safety, we fit independent Gaussian processes to the objective and constraint observations. The GP posterior for $f^\star$ induces lower and upper confidence bounds $\ell_n^f(x)$ and $u_n^f(x)$ such that the posterior mean lies inside $[\ell_n^f(x), u_n^f(x)]$ with, say, 95% probability; an analogous pair $\ell_n^g(x), u_n^g(x)$ is derived for $g^\star$.

These confidence bounds define conservative and optimistic estimates of the safe region,

$$\mathcal{S}_n = \{x : \ell_n^g(x) \geq 0\}, \qquad \hat{\mathcal{S}}_n = \{x : u_n^g(x) \geq 0\},$$

and identify candidate maximizers through

$$\mathcal{A}_n = \left\{ x \in \hat{\mathcal{S}}_n : u_n^f(x) \geq \max_{x' \in \mathcal{S}_n} \ell_n^f(x') \right\}.$$

The sample space is restricted to the provably safe set $\mathcal{S}_n$, whereas the target domain focuses on the optimistic maximizers $\mathcal{A}_n$. Applying ITL with $\mathcal{S} = \mathcal{S}_n$ guarantees that every query satisfies the safety constraint with high probability, because only points whose lower confidence bound remains non-negative are eligible. Simultaneously, setting $\mathcal{A} = \mathcal{A}_n$ targets those points most likely to improve the incumbent optimum—locations whose upper confidence bound could exceed the best certified value. Observe that the $\max_{x' \in \mathcal{S}_n} \ell_n^f(x')$ term is the best safe value known so far, so only points that might surpass it are included in $\mathcal{A}_n$.

Having adapted the transductive principle to a safe sequential design problem, we now turn to a complementary constraint: labeling must sometimes occur in *batches*. The underlying idea stays the same—identify a subset of examples whose labels will control the generalization error—but now the constraint is a fixed batch size rather than safety.

11.5 Batch active learning

Many labeling processes run in batches to leverage annotator availability. Suppose we are given an input domain $\mathcal{X}$ with distribution P over it, oracle access to an unknown function $f : \mathcal{X} \to \mathcal{Y}$, a finite population $X = \{x_1, \ldots, x_m\} \subseteq \mathcal{X}$, and a budget $b \leq m$. We must pick a subset $L \subseteq X$ of size b and request labels $\{f(x) : x \in L\}$. The goal is to choose L such that the learner trained on these labels generalizes well under P.

11.6 Auxiliary definitions

The batch formulation by Yehuda et al. (2022) relies on local homogeneity:

- For $x \in \mathcal{X}$ and $\delta > 0$, the ball $B_\delta(x) = \{x' \in \mathcal{X} : \|x - x'\| \leq \delta\}$ is *pure* if f is constant on that ball.

- For $L \subseteq \mathcal{X}$, define the covered region $C(L, \delta) = \bigcup_{x \in L} B_\delta(x)$. We write C_r for the subset on which the 1-nearest-neighbor classifier $\tilde{f}$ trained on $Z = \{(x, f(x)) : x \in L\}$ matches f, and C_w for the disagreement region.

- The impurity of radius δ is $\tilde{\pi}(\delta) = \mathbb{P}_{x \sim P}\big[B_\delta(x) \text{ is not pure}\big]$, a non-decreasing function of δ.

If a point is misclassified by the 1-NN classifier, it must either lie outside the covered region or inside a non-pure ball. Hence,

$$\mathbb{P}_{x \sim P}\big[\tilde{f}(x) \neq f(x)\big] \leq 1 - \mathbb{P}\big(C(L,\delta)\big) + \tilde{\pi}(\delta).$$

Active learning therefore seeks L and δ that minimize this upper bound. A practical strategy is to select δ first and then maximize the coverage probability $\mathbb{P}(C(L,\delta))$ under a budget constraint.

11.7 Coverage maximization and ProbCover

Given the bound above, we must solve

$$\max_{L \subseteq X,\, |L|=b} \mathbb{P}\Big(\bigcup_{x \in L} B_\delta(x)\Big),$$

but two obstacles arise: the underlying distribution P is unknown and the optimization problem is NP-hard. The fix proceeds in two steps:

1. Approximate P with the empirical distribution over X, which amounts to counting how many points lie inside the covered region.

2. Apply a greedy maximal-coverage heuristic known as PROBCOVER.

Let $G = (X, E)$ be a graph with edges between points at distance at most δ. PROBCOVER iteratively adds the point that covers the largest number of yet-uncovered neighbors:

$L \leftarrow \varnothing$
for $i = 1$ to b **do**
    $x^\star \leftarrow \arg\max_{x \in X} |\{x' : (x, x') \in E,\ x' \text{ not yet covered}\}|$
    $L \leftarrow L \cup \{x^\star\}$
    Remove all edges incident to points in $B_\delta(x^\star) \cap X$ (they need not be considered again)
end for
return L

This greedy routine provides a logarithmic approximation guarantee for set coverage problems and serves as a practical batch active learning policy: small δ focuses on fine-grained distinctions (lower impurity, smaller coverage), whereas large δ broadens coverage but risks mixing labels. By tuning δ and b, we trade off robustness of the 1-NN classifier against the number of labels queried.

11.8 Summary

We began by splitting the world into a target region and a feasible sampling set, quantifying information gain through ITL. We then reused this perspective to design a safe Bayesian optimization loop where feasibility is enforced through GP confidence bounds, and finally transitioned to batch selection by casting informativeness as probabilistic coverage.

12 Counterfactual Invariance

Machine-learning estimators often achieve high accuracy for the wrong reasons. Spurious correlations, confounders, and unobserved interventions can create shortcuts that models exploit during training but that fail under distribution shift. Counterfactual invariance addresses this failure mode by demanding that a predictor remain stable when nuisance factors are changed while the underlying causal mechanism of interest is held fixed. This chapter develops the idea systematically: starting from motivating pathologies, we introduce the causal-graph machinery necessary to reason about counterfactual statements, derive formal conditions for counterfactual invariance in both causal and anti-causal regimes, and finally discuss how to enforce those conditions in practice.

12.1 Motivation: when correlations lie

Classical statistical anecdotes already hint at the perils of naive correlation chasing:

- Observational studies once concluded that young children sleeping with a light on were more likely to develop myopia; later it was shown that myopic parents both left lights on and passed on their genetics.

- Correlations between mozzarella consumption and doctorates awarded, or between lice and health, are artifacts of shared confounders rather than direct causation.

- In medical imaging, Badgeley et al. found that a hip-fracture classifier had learned to use hospital-specific devices visible in X-rays rather than skeletal cues, leading to dramatic failures under domain shift.

These examples illustrate three recurring pitfalls: *reverse causation* (target causes features), *third-cause fallacies* (hidden confounders drive both feature and target), and *shortcut learning* (models latch onto non-causal proxies). Counterfactual invariance seeks representations and predictors that respond only to the causal content of the data.

12.2 Shortcut learning and domain shifts

Shortcut learning arises when causal and spurious factors co-occur in the training data. Given two environments $\mathcal{E}_1$ and $\mathcal{E}_2$ whose feature distributions differ via nuisance features W, an estimator f trained on $\mathcal{E}_1$ may exploit W as a proxy for the label Y rather than responding to the causal features X. At test time, a domain shift—either because W changes distribution or because we deploy f in a new environment—breaks this proxy, and accuracy collapses. The solution is to encode *invariant representations*: features of X that do not depend on the environment W yet retain the signal about Y.

12.3 Counterfactuals and invariance

Let X be a random feature vector representing an object, and let Y be the target variable we wish to predict. Suppose we also observe a random vector W capturing nuisance attributes: factors that may influence X but should not influence the prediction rule. A counterfactual $X(w)$ is the feature vector we would observe if we were to intervene and set W to the specific value w, leaving everything else unchanged. Formally, the intervention replaces the generative mechanism for W with the constant w while keeping the structural equations for other variables intact.

Definition 1 (Counterfactual invariance) *A predictor f is* counterfactually invariant *with respect to W if for any two values w, w' in the range of W and every realization of X we have*

$$f\big(X(w)\big) = f\big(X(w')\big).$$

In words, if we edit the nuisance factors while holding everything else constant, the prediction remains unchanged.

Intuitively, counterfactual invariance demands that f ignore all pathways through which W can influence the prediction except via the causal content shared across interventions.

12.4 Two causal regimes

The causal structure of the problem determines how we should enforce invariance. Two canonical regimes cover most applications:

- **Causal regime.** Changing features X has a causal effect on Y, e.g., in a medical-diagnosis setting where symptoms influence the disease classification. Here we typically assume $X \to Y$.

- **Anti-causal regime.** The target Y causes the features X, e.g., in disease detection where the disease (cause) produces observable symptoms (effects). Here we model $Y \to X$.

In both regimes the nuisance W may influence X but must not affect Y. The difference lies in whether Y lies downstream or upstream of X in the causal graph, which alters the conditional independences we can exploit.

12.5 Causal graphs and d-separation

To reason formally, we represent the relationships among W, X, Y, and latent variables through a causal graph. Nodes correspond to random variables, directed edges denote causal influence, and we assume the graph is a directed acyclic graph (DAG). For example, in the causal regime (where $X \to Y$) a minimal graph might include:

$$W \to X \to Y,$$

with potential latent confounders U influencing both W and X, or selection variables S governing which samples appear in the dataset. In the anti-causal regime, the edge direction flips: $Y \to X$, yet W still influences X.

Conditional independences implied by the graph are characterized via *d-separation*. A path between two nodes is blocked if it contains:

- A chain $A \to B \to C$, $A \leftarrow B \to C$, or $A \leftarrow B \leftarrow C$ where the middle node B is conditioned on.

- A collider $A \to B \leftarrow C$ where the middle node B is *not* conditioned on (nor any of its descendants).

If every path between two nodes is blocked given a conditioning set Z, the nodes are d-separated, implying conditional independence. This rule lets us derive necessary independences for counterfactual invariance.

Intuition. D-separation tracks whether there is still an "open pipeline" along which information can travel. In a chain or fork, every influence flowing from one endpoint to the other must pass through the middle node, so once we observe that node the pipeline is saturated—learning B already captures everything A could reveal about C. A practical fork example is a genetic mutation B that simultaneously raises cholesterol A and blood pressure C: before measuring the gene, high cholesterol hints at high blood pressure; after conditioning on the mutation, the endpoints decouple because their only link has been cut. Likewise, in the chain $C \to B \to A$ (smoking $\to$ tar buildup $\to$ chronic cough) conditioning on tar levels breaks the association between smoking and cough, because tar already summarizes whatever smoking would have conveyed. Colliders behave oppositely: two independent causes $A \to B \leftarrow C$ remain disconnected unless we observe the collision (e.g., conditioning on sneezing that can be caused by both a cold and allergies), in which case the path becomes active and learning one cause "explains away" the other.

12.6 Confounding and selection pitfalls

Two additional phenomena can break counterfactual invariance if unaddressed:

- **Confounding.** Hidden variables U that influence both W and X (or Y) create associations that do not disappear under interventions. Simpson's paradox—where aggregated data reverses the trend observed within subgroups—is a textbook manifestation.

- **Selection bias.** Conditioning on a selection variable S that depends on W and X introduces collider bias: even independent variables become correlated when we restrict attention to samples with $S = 1$ (e.g., only applicants on LinkedIn).

Accounting for these effects requires either modeling the hidden variables explicitly or designing algorithms that enforce the necessary conditional independences despite selection.

12.7 Necessary conditions for invariance

Counterfactual invariance imposes specific conditional independence relations on $f(X)$. Intuitively, if f ignores W even under interventions, then after conditioning on the causal parents, the prediction cannot leak information about W. The following conditions are necessary:

- **Anti-causal scenario:** $f(X) \perp W \mid Y$. Once we know the true label Y, the prediction $f(X)$ must reveal nothing about the nuisance W. Otherwise, altering W while holding Y fixed would change f.

- **Causal scenario without selection:** $f(X) \perp W$. Because Y is downstream of X, any dependence on W indicates that f has retained nuisance information.

- **Causal scenario with selection bias:** If selection variables S depend on both W and X, then colliders induced by conditioning on $S = 1$ can reintroduce spurious correlations. In this case, a conservative requirement is $f(X) \perp W \mid Y$, provided that Y does not depend on the selection mechanism once X and W are fixed.

Below we sketch why these conditions follow from d-separation.

Anti-causal proof sketch. Assume $Y \to X$ and $W \to X$, possibly with confounders capturing unobserved causes of W and Y. Counterfactual invariance means $f(X)$ only depends on the portion of X that remains invariant under interventions on W, typically written as $X_W^\perp$. In the anti-causal graph, every path from $X_W^\perp$ to W is blocked once we condition on Y: colliders opened by Y are observed, while colliders through latent variables remain unobserved and thus block the path. Therefore, if f only uses $X_W^\perp$, we must have $f(X) \perp W \mid Y$.
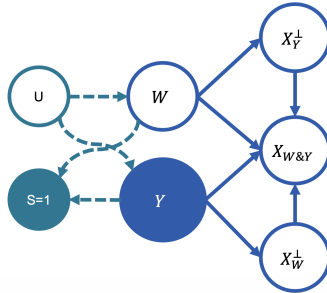

Figure 2: Anti-causal regime

Causal proof sketch. When $X \to Y$, f can depend on X directly. Yet to be invariant, f must ignore all components of X that are influenced by W. In the absence of selection, the graph ensures that every path from $X_W^\perp$ to W goes through a collider—either $X_{W\&Y}$ or Y itself—that is not observed. Consequently, the residual representation cannot correlate with W, leading to the unconditional independence $f(X) \perp W$. If we condition on a selection variable, some paths may become unblocked, so we revert to the more cautious requirement conditioned on Y.

12.8 Enforcing invariance via distribution matching

The independence conditions can be operationalized by matching the distributions of the predictions across environments. Consider the anti-causal setting with binary W and Y. Counterfactual
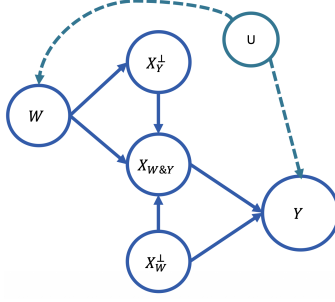
Figure 3: Causal regime

invariance demands

$$f(X) \mid \{W = w, \, Y = y\} \quad \overset{d}{=} \quad f(X) \mid \{W = w', \, Y = y\}$$

for all w, w', y. In practice we can:

1. Choose a discrepancy measure $\Delta(p, q)$ between probability distributions (e.g., maximum mean discrepancy, Wasserstein distance, or KL divergence).

2. Add a regularization term that penalizes discrepancies between the empirical distributions of $f(X)$ across the nuisance groups conditioned on Y:

$$\mathcal{L}_{\mathrm{inv}} = \sum_{y} \sum_{w,w'} \Delta\Big(\hat{p}\big(f(X) \mid W = w, Y = y\big), \, \hat{p}\big(f(X) \mid W = w', Y = y\big)\Big).$$

3. Optimize the predictor to minimize the original task loss plus $\lambda\mathcal{L}_{\mathrm{inv}}$, where λ balances task fit and invariance.

This strategy enforces that, within each class y, the representation used by f carries no information about the nuisance. Extensions include adversarial training where a discriminator tries to predict W from the learned representation while the encoder attempts to fool it, or explicit data augmentation with synthetically generated counterfactuals when such interventions are available.

12.9   Summary

Counterfactual invariance provides a principled remedy for shortcut learning. By explicitly modeling the nuisance factors W, constructing counterfactuals $X(w)$, and reading off the necessary conditional independences from causal graphs, we can reason about when a predictor truly focuses on causal features. Enforcing these independences—through distribution matching, adversarial debiasing, or counterfactual augmentation—yields models whose predictions remain stable even when environments shift. In domains where decisions must rest on the right reasons, such invariance is not merely desirable; it is essential.

13   Variational Autoencoders

Learning useful representations without labels requires us to tell the model what "useful" means. Variational autoencoders (VAEs) formalize this request by combining the information-theoretic appeal of autoencoders with explicit probabilistic modeling. This chapter develops VAEs from the bottom up: we motivate the desiderata for unsupervised representations, examine why maximizing mutual information alone fails, cast autoencoders in a Bayesian light, derive the evidence lower bound (ELBO), and describe how to optimize it in practice.

13.1   Representation learning without supervision

Deep networks can be viewed as compositions of an *encoder* that maps the input $x \in \mathcal{X}$ into a latent representation z and a *decoder* (or predictor) that turns z into a task output. When no labels are available, the only supervision we can provide is the input itself. Good representations should therefore satisfy three properties:

Informative. The original input should be recoverable from z, so no essential information is discarded.

Disentangled. Individual coordinates of z should align with distinct generative factors (pose vs. lighting, stroke width vs. digit identity, *etc.*), enabling controlled manipulation.

Robust. Small perturbations in the input should not cause drastic changes in z, and conversely modifying z slightly should not flip the reconstruction arbitrarily.

Autoencoders optimize informativeness by minimizing reconstruction error, but disentanglement and robustness typically require further regularization.

13.2   The infomax principle and its limitations

Let $\text{enc}_\theta \in \mathcal{H}$ be a (possibly stochastic) encoder. The infomax principle (Linsker, 1988) advocates selecting the parameters θ that maximize the mutual information $I(X; Z)$ between inputs X and their representations $Z = \text{enc}_\theta(X)$:

$$I(X; Z) = \int p(x, z) \log \frac{p(x, z)}{p(x)\, p(z)}\, dx\, dz$$

With only data $x_1, \ldots, x_n$, the expectation can be approximated via Monte Carlo:

$$I(X; Z) \approx \sum_{i \leq n} \mathbb{E}_{Z|x_i} \left[\log p\left(x_i \mid Z\right) \right]$$

which resembles the reconstruction score of an autoencoder. Unfortunately, maximizing $I(X; Z)$ alone can be trivial whenever $\mathcal{X}$ and $\mathcal{Z}$ are rich enough: the encoder can simply implement a bijection (or memorize the dataset) so that Z is a lossless copy of X. Such degenerate solutions are perfectly informative but neither disentangled nor robust, underscoring the need for additional inductive biases.

13.3   Latent-variable generative modeling

To go beyond empty informativeness, we posit a generative story for the data. A latent variable Z is sampled from a prior $p(z)$, and the observation X is drawn from a conditional decoder $p_\theta(x \mid z)$. This *decoder* can be a deep neural network that outputs the parameters of a likelihood distribution (Gaussian for continuous data, Bernoulli/Categorical for binary or discrete data). High-quality representations now correspond to posterior inferences: given x, infer the distribution $p_\theta(z \mid x)$ of latent causes. Integrating the latent variable out gives the marginal likelihood $p_\theta(x) = \int p_\theta(x \mid z)p(z)\, dz$, which quantifies how well the generative model explains the observation.

Two conceptual benefits arise:

- Sampling new x is straightforward: draw $z \sim p(z)$ and decode it. VAEs thus act as generative models capable of "hallucinating" plausible data such as handwritten digits or faces.

- Regularization becomes Bayesian: the prior $p(z)$ codifies which representations are plausible (e.g., standard normal means centered, isotropic latent factors), discouraging memorization.

The main obstacle is posterior inference. Computing $p_\theta(z \mid x)$ exactly is typically intractable because it requires normalizing the product $p_\theta(x \mid z)p(z)$ over all z. Variational inference circumvents this by introducing an auxiliary encoder $q_\phi(z \mid x)$, parameterized by ϕ, that approximates the true posterior.

13.4   A manifold perspective

High-dimensional data often concentrate near a *manifold*—a smooth, low-dimensional surface embedded in the ambient space. Pictures of faces, for instance, span thousands of pixels yet vary along a handful of semantic axes such as pose, lighting, and expression. VAEs can be interpreted as tools for learning coordinates on this manifold. The decoder $p_\theta(x \mid z)$ acts like a chart that maps latent coordinates z into points on (or near) the data manifold, while the encoder $q_\phi(z \mid x)$ performs the inverse mapping from observations back to manifold coordinates. The prior $p(z)$ then regularizes the geometry of this manifold by favoring latents near the origin, which in turn encourages smooth, disentangled directions. Thinking in manifold terms clarifies why VAEs produce interpolations that remain realistic: straight lines in latent space correspond to geodesic-like curves on the learned manifold, yielding gradual transitions in pixel space.
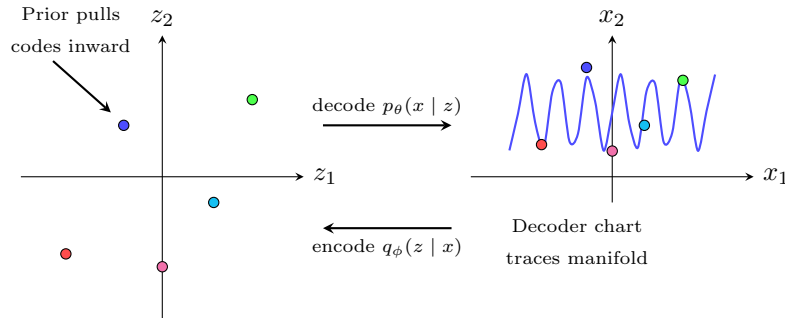

Figure 4: The encoder learns manifold coordinates while the decoder maps latent points back onto the curved data manifold, keeping interpolations smooth.

13.5   A Bayesian detour

Bayesian estimation provides intuition for why priors help disentangle noise from signal. Suppose we wish to estimate the mean shoe size μ in a small town. A Gaussian prior $\mu \sim \mathcal{N}(\alpha, \beta)$ captures our belief before collecting data. Observing measurements $x_1, \ldots, x_n$ with $x_i \sim \mathcal{N}(\mu, 1)$ yields a Gaussian posterior $\mathcal{N}(\mu \mid m, s^2)$ that balances the prior and the evidence. When few samples are available, the posterior stays near α (robustness to noise); as n grows, the data dominate (informativeness). VAEs apply the same logic in every latent dimension: the prior $p(z)$ nudges encodings toward structured, disentangled regions while still allowing the decoder to reconstruct the input accurately.

14   Non-parametric Bayesian Methods

How can we endow probabilistic models with enough flexibility to grow with the data instead of committing to a fixed number of parameters? Non-parametric Bayesian methods answer this question by replacing finite-dimensional priors with distributions over infinite-dimensional objects such as probability measures. This chapter traces the path from Bayesian inference for a single Gaussian to Gaussian mixture models (GMMs) with an unbounded number of clusters, highlighting the role of conjugate priors, Gibbs sampling, Dirichlet processes, and exchangeability.

14.1   Warm-up: Bayesian inference for a single Gaussian

Consider one-dimensional observations $X = \{x_1, \ldots, x_n\}$ drawn i.i.d. from $\mathcal{N}(\mu, \sigma^2)$ with known variance $\sigma^2 = 1$ but unknown mean μ. Bayesians update a prior belief $\mu \sim \mathcal{N}(m_0, k_0^{-1})$ via Bayes' rule:

$$p(\mu \mid X) \propto p(X \mid \mu)\, p(\mu) = \left(\prod_{i=1}^{n} \mathcal{N}(x_i \mid \mu, 1) \right) \mathcal{N}(\mu \mid m_0, k_0^{-1}).$$

Because Gaussians are conjugate to themselves, the posterior remains Gaussian with parameters

$$m_n = \frac{k_0 m_0 + n\bar{x}}{k_0 + n}, \qquad k_n = k_0 + n,$$

where $\bar{x}$ is the sample mean. Intuitively, k_0 quantifies how confident we were in m_0; after seeing n data points, the precision simply adds up. The posterior mean m_n is a weighted average between the prior mean and the data mean, showcasing regularization against outliers and a principled notion of uncertainty.

14.2   Multivariate Gaussians and conjugate priors

For d-dimensional data $x_i \sim \mathcal{N}(\mu, \Sigma)$ with both μ and Σ unknown, the normal-inverse-Wishart (NIW) distribution is a conjugate prior whose hyperparameters carry clear meaning:

- $m_0 \in \mathbb{R}^d$: prior mean vector,

- $k_0 > 0$: scaling factor controlling confidence in m_0,

- $S_0 \in \mathbb{R}^{d \times d}$: scale matrix encoding prior beliefs about covariance structure,

- $\nu_0 > d - 1$: degrees of freedom, controlling confidence in S_0.

Crucially, the inverse-Wishart component enforces that sampled covariance matrices remain positive semi-definite, which is required for any valid multivariate Gaussian. The NIW density is

$$p(\mu, \Sigma) = \text{NIW}(\mu, \Sigma \mid m_0, k_0, S_0, \nu_0).$$

Given a dataset X, Bayes' rule yields the posterior

$$p(\mu, \Sigma \mid X) = \text{NIW}(m_n, k_n, S_n, \nu_n),$$

with updates

$$k_n = k_0 + n, \qquad \nu_n = \nu_0 + n,$$
$$m_n = \frac{k_0 m_0 + n\bar{x}}{k_0 + n}, \qquad S_n = S_0 + S_X + \frac{k_0 n}{k_0 + n}(\bar{x} - m_0)(\bar{x} - m_0)^{\top},$$

where S_X is the sample covariance matrix. Only sufficient statistics (mean and covariance) are required, making posterior updates computationally light even for large n.

14.3 Sampling with semi-conjugate priors

Fully conjugate priors are convenient but sometimes too rigid—for instance, we may want to express separate beliefs about the location and spread of the data. Under an NIW prior the strength of the prior mean and the tightness of the covariance are linked through k_0, so tightening one automatically tightens the other. Semi-conjugate priors break this coupling while keeping *conditionally* conjugate updates, which is all Gibbs sampling needs. A typical choice specifies

$$\mu \sim \mathcal{N}(m_0, V_0), \qquad \Sigma \sim \mathrm{IW}(S_0, \nu_0),$$

so the joint density no longer has a closed form but the conditional posteriors do. The Given current samples, Gibbs sampling iterates:

$$\mu \mid \Sigma, X \sim \mathcal{N}(m_p, V_p)$$

$$\Sigma \mid \mu, X \sim \mathrm{IW}(S_p, v_p)$$

Each step has the same flavor as the fully conjugate update: the data provide sufficient statistics, while the prior contributes virtual observations. Even though the joint posterior $p(\mu, \Sigma \mid X)$ cannot be written explicitly, the Markov chain that alternates these two conditional draws converges to it under mild conditions. Practical samplers further exploit graphical- model independencies (via d-separation) and Rao-Blackwellization to reduce variance and shorten burn-in.

Gibbs sampler details. A single Gibbs sweep for this semi-conjugate model proceeds as follows:

1. Given the current covariance sample $\Sigma^{(t)}$, draw a new mean by sampling $\mu^{(t+1)}$ from the Gaussian conditional above (using $\Sigma^{(t)}$ inside the formula).

2. Plug $\mu^{(t+1)}$ into the inverse-Wishart conditional to draw the next covariance sample $\Sigma^{(t+1)}$.

3. Repeat these two steps for many iterations, discarding the first T_{burn} draws as burn-in and optionally thinning the remainder.

Because each conditional depends on the latest value of the other block, the chain "zig-zags" through the (μ, Σ) space but still converges to the true joint posterior. Conditional independence structure (e.g., between clusters in a mixture) allows updating blocks in parallel or integrating out nuisance variables before running the sampler.
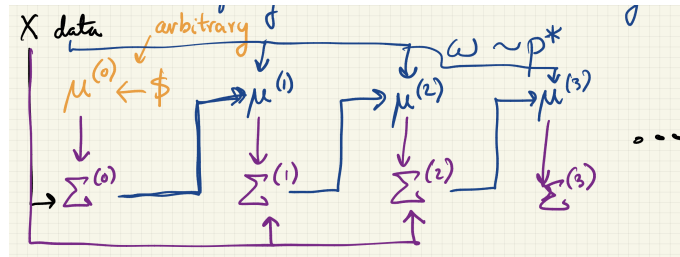

Figure 5: Gibbs sampling illustration.

14.4 Gibbs sampling in a nutshell

Whenever the posterior $p(\Theta \mid X)$ factorizes into conditionals that are easy to sample from, *Gibbs sampling* provides a route to approximate inference even if the joint density is intractable. Let $\Theta = (\Theta_1, \ldots, \Theta_\ell)$ be the latent variables of interest. Gibbs sampling constructs a Markov chain $\{\Theta^{(t)}\}_{t=0}^{\infty}$ by iterating:

1. Initialize $\Theta^{(0)}$ arbitrarily (e.g., random cluster assignments).

2. For $t = 0, 1, 2, \ldots$:

   (a) Sample $\Theta_1^{(t+1)} \sim p(\Theta_1 \mid \Theta_2^{(t)}, \ldots, \Theta_\ell^{(t)}, X)$.

(b) Sample $\Theta_2^{(t+1)} \sim p(\Theta_2 \mid \Theta_1^{(t+1)}, \Theta_3^{(t)}, \ldots, \Theta_\ell^{(t)}, X)$.

(c) Continue cycling through all coordinates until $\Theta_\ell^{(t+1)}$ is sampled given the most recent values of the others.

Each conditional draw is typically conjugate (or otherwise tractable) because all variables except one are held fixed. Under mild regularity conditions this Markov chain has $p(\Theta \mid X)$ as its stationary distribution, so samples collected after a burn-in period approximate the true posterior. In practice we discard the first T_{burn} iterations, then thin or average the remaining ones to estimate expectations. The method shines in models such as GMMs where conditionals like $p(z_i \mid z_{-i}, \pi, \mu, \Sigma, X)$ or $p(\mu_k, \Sigma_k \mid z, X)$ admit closed forms.
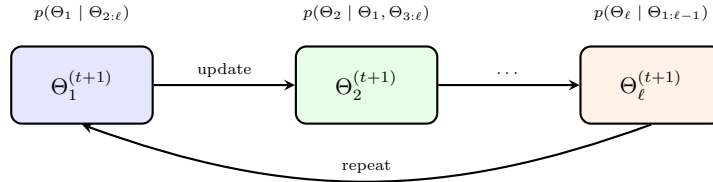

Figure 6: Gibbs Sampling

15 Probably Approximately Correct Learning

Probably Approximately Correct (PAC) learning formalizes the intuition that a model should capture the regularities in the data-generating process, not just the accidents of a finite dataset. The framework asks how much data a learning algorithm needs before the hypothesis it outputs generalizes, and it does so without assuming a particular distribution for the inputs.

15.1 From Empirical Patterns to Guarantees

Every supervised learning pipeline starts from a labeled sample $Z = \{(x_i, y_i)\}_{i=1}^n$ drawn from an unknown distribution. Fitting a hypothesis that performs well on Z is easy; guaranteeing that the same hypothesis will predict unseen examples demands more structure. Statistical learning theory introduces the *generalization error*

$$R(\hat{c}) = \mathbb{P}_{X \sim D}\big(\hat{c}(X) \neq c(X)\big),$$

which measures how often hypothesis $\hat{c}$ disagrees with the (unknown) target concept c under the true distribution D on the instance space $\mathcal{X}$. Because D and c are inaccessible, the learner minimizes the empirical error

$$\widehat{R}_n(\hat{c}) = \frac{1}{n} \sum_{i=1}^n \mathbb{I}\{\hat{c}(x_i) \neq y_i\},$$

hoping that $\widehat{R}_n$ is a faithful estimate of R. PAC learning quantifies when this hope is justified.

15.2 Instance, Concept, and Hypothesis Spaces

An *instance space* $\mathcal{X}$ enumerates all objects the learner may observe. A *concept* c is a subset of $\mathcal{X}$ or, equivalently, a label function $c : \mathcal{X} \to \{0, 1\}$. A *concept class* $\mathcal{C}$ collects candidate targets, whereas a *hypothesis class* $\mathcal{H}$ contains the functions the algorithm is allowed to output. The learner receives Z with labels $y_i = c(x_i)$ (realizable setting) or draws from a distribution on $\mathcal{X} \times \{0, 1\}$ (agnostic setting) and must pick $\hat{c} \in \mathcal{H}$ that approximates c.

15.3 The PAC Criterion

A learning algorithm A is a PAC learner for concept class $\mathcal{C}$ if there exists a polynomial p such that for any distribution D on $\mathcal{X}$, any tolerance parameters $0 < \varepsilon, \delta < 1/2$, and any target $c \in \mathcal{C}$, the algorithm produces $\hat{c} \in \mathcal{H}$ satisfying

$$\mathbb{P}_{Z \sim D^n} \left(R(\hat{c}) \leq \varepsilon \right) \geq 1 - \delta$$

whenever it receives at least $n \geq p(1/\varepsilon, 1/\delta, \text{size}(c))$ samples. The polynomial captures *sample complexity*; ε controls accuracy, and δ controls confidence. Efficient PAC learners also run in time polynomial in the same quantities.

15.4 Axis-Aligned Rectangles as a Running Example

To build intuition, consider $\mathcal{X} = \mathbb{R}^2$ and let $\mathcal{C}$ be all axis-aligned rectangles. Given positive and negative labeled points, a natural learner outputs $\hat{R}$, the smallest rectangle containing every positive sample.

Intuitively, this rectangle expands just enough to explain the data, so it should not misclassify too many points outside the observed cloud. The theory turns this intuition into a guarantee.

Partition the true rectangle R into four thin *strips*: upper, lower, left, and right, each with probability mass $\varepsilon/4$ under D. Let the event $\hat{R}_{\text{IG}}$ denote that the learned rectangle intersects all strips ("IG" for *is good*). If a strip is missed, that entire portion of R will be falsely labeled negative, contributing at least $\varepsilon/4$ error. Conversely, intersecting every strip ensures that the area where $\hat{R}$ differs from R has probability at most ε.

The key insight is that if the learned rectangle $\hat{R}$ misses any strip, it will incur large error. We formalize this by analyzing the probability that $\hat{R}$ intersects all four strips.
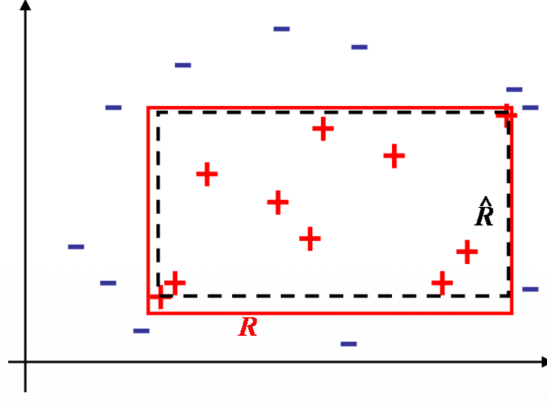
Figure 7: Axis-aligned rectangle learning: the smallest rectangle $\hat{R}$ containing all positive samples.
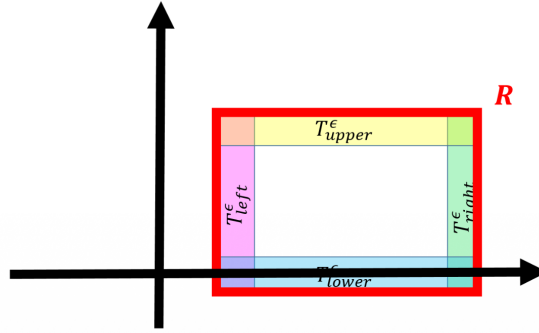

Figure 8: Partitioning the true rectangle into four strips, each with probability mass $\varepsilon/4$.

Derivation. The chance a fixed strip receives no sample is $(1 - \varepsilon/4)^n \leq \exp(-n\varepsilon/4)$ by $1 + x \leq e^x$. A union bound over the four strips yields

$$\mathbb{P}\left(\hat{R}_{\text{IG}}\right) \geq 1 - 4\exp\left(-\frac{n\varepsilon}{4}\right).$$

On $\hat{R}_{\text{IG}}$, the symmetric difference $R \triangle \hat{R}$ has probability at most ε, so

$$\mathbb{P}\left(R(\hat{R}) \leq \varepsilon\right) \geq \mathbb{P}\left(\hat{R}_{\text{IG}}\right).$$

Intuitively, intersecting every strip forces $\hat{R}$ to stretch all the way to each face of the true rectangle: if it were to stop short (say, at the top), it would exclude the sample that witnessed that strip, contradicting minimality. Consequently the only region where R and $\hat{R}$ can disagree is confined to the four strips themselves, whose total probability mass is at most ε. Bounding the disagreement set is therefore equivalent to bounding $R(\hat{R})$.

It suffices to set $n \geq \frac{4}{\varepsilon}\log\frac{4}{\delta}$ to make the failure probability at most δ. The dependence is logarithmic in $1/\delta$ and linear in $1/\varepsilon$, exactly the scaling promised by the PAC definition.

15.5   Induction Principles and Empirical Risk Minimization

Learning can be viewed as an induction principle: from observed labeled samples we induce a rule that will be used to classify new points. Once a hypothesis has been induced, deduction applies the rule to unseen inputs, while *transduction* skips explicit model building and predicts labels for a specific test set directly. PAC learning focuses on induction and the conditions under which it is justified.

The standard induction rule is *empirical risk minimization* (ERM). Given a hypothesis class $\mathcal{C}$,

ERM selects
$$\hat{c}_n^* \in \arg\min_{c \in \mathcal{C}} \widehat{R}_n(c),$$

where $\widehat{R}_n(c)$ is the empirical classification error. This choice is computable without any prior assumptions on D, but it shifts the burden to analysis: we need distribution-independent bounds on the excess risk
$$R(\hat{c}_n^*) - \inf_{c \in \mathcal{C}} R(c).$$

The rest of the chapter develops tools to control this deviation with high probability.

15.6 Uniform Convergence and the VC Inequality

The empirical minimizer is data-dependent, so the law of large numbers for a *fixed* classifier does not directly apply. The remedy is *uniform convergence*: ensure that *all* hypotheses in $\mathcal{C}$ have empirical risks close to their true risks simultaneously.

Let $c^\star = \arg\min_{c \in \mathcal{C}} R(c)$ denote the best-in-class classifier. Then
$$R(\hat{c}_n^*) - R(c^\star) \leq \left| R(\hat{c}_n^*) - \widehat{R}_n(\hat{c}_n^*) \right| + \left| \widehat{R}_n(c^\star) - R(c^\star) \right|$$
$$\leq 2 \sup_{c \in \mathcal{C}} \left| \widehat{R}_n(c) - R(c) \right|.$$

Consequently,
$$\mathbb{P}\left(R(\hat{c}_n^*) - \inf_{c \in \mathcal{C}} R(c) > \varepsilon \right) \leq \mathbb{P}\left(\sup_{c \in \mathcal{C}} \left| \widehat{R}_n(c) - R(c) \right| > \varepsilon/2 \right).$$

This bound explains why empirical minimizers can have smaller *empirical* error than the true minimizer $c^\star$ yet still generalize: what matters is the uniform deviation between empirical and true risks across the class.

15.7 Finite Hypothesis Classes and Consistency

For a finite hypothesis class $\mathcal{H}$ with $N = |\mathcal{H}|$, concentration inequalities make uniform convergence explicit. For any fixed h, the empirical risk is an average of Bernoulli errors in $[0, 1]$, so Hoeffding's inequality gives
$$\mathbb{P}\left(\left| \widehat{R}_n(h) - R(h) \right| > \varepsilon \right) \leq 2 \exp(-2n\varepsilon^2).$$

Applying a union bound over N hypotheses yields
$$\mathbb{P}\left(\sup_{h \in \mathcal{H}} \left| \widehat{R}_n(h) - R(h) \right| > \varepsilon \right) \leq 2N \exp(-2n\varepsilon^2).$$

Equivalently, with probability at least $1 - \delta$, the following *uniform confidence interval* holds for all $h \in \mathcal{H}$:
$$R(h) \leq \widehat{R}_n(h) + \sqrt{\frac{\log N + \log(2/\delta)}{2n}}.$$

The variance term decays as $1/\sqrt{n}$ and grows only logarithmically with N, which makes large but finite hypothesis classes viable.

In the realizable case, where the algorithm can return a hypothesis consistent with the sample ($\widehat{R}_n(\hat{c}) = 0$), a sharper argument is possible. Every $h \in \mathcal{H}$ with true error $R(h) > \varepsilon$ must mislabel at least an ε-fraction of the instance space, so the probability that none of the n samples expose its error region is at most $\exp(-n\varepsilon)$. A union bound over all N bad hypotheses yields
$$\mathbb{P}\left(R(\hat{c}) > \varepsilon \right) \leq |\mathcal{H}| \exp(-n\varepsilon).$$

Equivalently, for any $\delta > 0$, it suffices to take
$$n \geq \frac{1}{\varepsilon}\left(\log|\mathcal{H}| + \log\frac{1}{\delta} \right)$$

to guarantee $R(\hat{c}) \leq \varepsilon$ with probability at least $1 - \delta$. The logarithmic dependence on $|\mathcal{H}|$ demonstrates that even exponentially large hypothesis spaces can be learnable, provided the algorithm searches them effectively.

15.8 Agnostic PAC Learning and Bayes Risk

The realizable assumption breaks when identical feature vectors carry different labels, as in noisy measurement regimes. In this *agnostic* scenario the benchmark is the Bayes optimal classifier $c_{\text{Bayes}}(x) = \arg\max_{y \in \{0,1\}} \mathbb{P}(y \mid x)$ with minimal achievable risk $R^\star$. Unlike the realizable case, no algorithm can guarantee error below $R^\star$ if the true labeling rule lies outside the hypothesis class. Instead we ask the learner to track the best rule available inside $\mathcal{C}$.

A hypothesis class $\mathcal{C}$ is agnostically PAC learnable if there exists A that, for every distribution on $\mathcal{X} \times \{0, 1\}$ and every ε, δ, outputs $\hat{c}$ with

$$\mathbb{P}\left(R(\hat{c}) - \inf_{c \in \mathcal{C}} R(c) \leq \varepsilon \right) \geq 1 - \delta.$$

The learner now competes with the best function inside $\mathcal{C}$ rather than with the Bayes classifier itself. This shift has two consequences: (i) the comparison point is $R_{\mathcal{C}}^\star := \inf_{c \in \mathcal{C}} R(c)$, the approximation error induced by the hypothesis class, and (ii) the sample complexity typically increases because the learner must control both estimation error (difference between $\hat{c}$ and the empirical minimizer) and approximation error (difference between $R_{\mathcal{C}}^\star$ and the true Bayes risk). The guarantee above quantifies how many samples suffice to push the estimation error down to ε, whatever the irreducible noise may be.

15.9 Controlling Complexity via VC Dimension

For infinite hypothesis classes, cardinality-based bounds break down: plugging $|\mathcal{H}| = \infty$ into the finite-class estimate yields no information, and even countably infinite classes defeat a naive union bound because one must sum probabilities over infinitely many "bad" hypotheses. The *Vapnik–Chervonenkis (VC) dimension* provides a refined measure of capacity in this regime. Shattering captures the idea of flexibility: a set $A \subseteq \mathcal{X}$ is *shattered* by $\mathcal{C}$ if every possible labeling of A can be realized by some $c \in \mathcal{C}$. The VC dimension $\text{VC}_{\mathcal{C}}$ is the size of the largest shattered set.

Examples give intuition about how geometry restricts shattering power:

- Any two points in $\mathbb{R}^2$ can be shattered by axis-aligned rectangles.

- Intervals on the real line shatter any two points but not all triples.

- Certain triples in $\mathbb{R}^3$ are shattered by rectangles, but not every triple, placing $\text{VC}_{\mathcal{C}}$ between two and three.

Finite VC dimension ensures that empirical risk minimization cannot overfit arbitrarily: the number of distinct labelings induced on any sample grows polynomially with n, so concentration inequalities still control the gap between empirical and true risk. Formally, if $\text{VC}_{\mathcal{C}} < \infty$, empirical risk minimization achieves

$$\mathbb{P}\left(R(\hat{c}_n^*) - \inf_{c \in \mathcal{C}} R(c) > \varepsilon \right) \leq 9 n^{\text{VC}_{\mathcal{C}}} \exp\left(-\frac{n\varepsilon^2}{32} \right),$$

which tends to zero as n grows. Thus finite VC dimension is both necessary and sufficient for PAC learnability in many settings.

15.10 Empirical Risk Minimization for Hyperplanes

Linear separators provide a concrete example of an infinite hypothesis class that is still learnable. Let $\mathcal{C}$ be the set of all hyperplanes in $\mathbb{R}^d$, which is uncountable. However, on a fixed sample of size n, only finitely many distinct labelings are possible. A *fingering* argument makes this explicit: choose any d sample points in general position (which holds with probability one under a density), and the hyperplane passing through them defines two classifiers depending on which side is labeled positive. There are at most $2\binom{n}{d}$ such classifiers, so the effective size of the class on the sample is polynomial in n for fixed d.

Moreover, for any linear classifier c there exists a hyperplane through d sample points whose empirical error differs by at most d/n. The intuition is that two hyperplanes can agree on all

but the points lying on the separating plane; those are at most d points. This reduces ERM for hyperplanes to ERM over a finite class of size $2\binom{n}{d}$, so uniform convergence still holds. In particular, for $n \geq d$ and $\varepsilon \geq 2d/n$, one obtains bounds of the form

$$\mathbb{P}\left(R(\hat{c}) > \inf_{c \in \mathcal{C}} R(c) + \varepsilon\right) \leq 2\binom{n}{d}\exp\left(-\frac{n\varepsilon^2}{2}\right),$$

illustrating a polynomial prefactor in n^d and an exponential decay in $n\varepsilon^2$. For fixed dimension d, the deviation probability still vanishes rapidly as n grows, confirming learnability without distributional assumptions.

If the best linear classifier has zero error ($R(c^\star) = 0$), the convergence rate improves: the probability of an ε-bad classifier scales like

$$\mathbb{P}\left(R(\hat{c}_n) > \varepsilon\right) \leq 2\binom{n}{d}\exp\left(-\varepsilon(n-d)\right),$$

since errors can occur only on the $n - d$ points not used to define the separating hyperplane. This yields an $\exp(-n\varepsilon)$ dependence rather than $\exp(-n\varepsilon^2)$.

Note. Finding the optimal dichotomy of n labeled points in $\mathbb{R}^d$ is NP-hard in the worst case, so ERM for hyperplanes can be computationally difficult even though it is statistically learnable.

15.11 Strong and Weak Learning Perspectives

The PAC condition can be relaxed to *weak learning*: instead of demanding arbitrarily small ε, one requires $R(\hat{c}) \leq \frac{1}{2} - \gamma$ for some fixed $\gamma > 0$ with high probability. Weak learners are tractable building blocks for ensemble methods such as boosting, which combine many slightly better-than-random hypotheses into a strongly consistent classifier. In contrast, strong PAC learning ensures (ε, δ) guarantees for every positive pair. Both notions rely on the same vocabulary—generalization error, sample complexity, and hypothesis class capacity—and highlight the central PAC message: learnability hinges on balancing expressive power with the ability to control error uniformly over the data-generating process.